60453000 F

CONTROL DATA CORPORATION

TAF/TS
VERSION 1
REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 1

ALPHABETICAL COMMAND SUMMARY

BEGIN Request 2-14
BTRAN Request 5-1
BWAITINP Request 2-14

CALLRTN Request 2-4
CALLTSK Request 2-5
CEASE Request 2-16
CMDUMP Request 6-4

DSDUMP Request 6-1

EXTRACT Request 2-19

IIO Request 2-13
INSERT Request 2-14
ITL Request 2-16

JOURNL Request 4-4

K.DSDUMP Command 6-4
K.DUMP Command 6-5
K.DUMPLIM Command 6-5
KPOINT Request 2-17
KTSDMP Request 6-5

LIBTASK Control Statement 9-2
LOADCB Request 2-9
LOGIN Request 2-18
LOGT Request 2-17

NEWTRAN Request 2-6

RECALAL Request 2-16
RELSCB Request 2-11

SEND Request 2-9
STRAN Macro 10-2
SUBMT Request 5-3

TARO Request 2-12
TIMCNT Macro 10-1
TPSTAT Request 2-14
TRAN Macro 10-1
TRANCHK Request 2-6
TSIM Request 2-11

VALNET Statement 8-2

WAIT Request 2-18
WAITINP Request 2-7

60453000

CONTROL DATA CORPORATION

TAF/TS
VERSION 1
REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 1

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released for NOS 1.2 and TAF 1.0 at PSR level 439.
(11-76)	
B	Revised to reflect NOS 1.2 at PSR level 454 and to make typographical and technical corrections.
(08-04-77)	All error messages are reformatted and have moved to appendix D. Support of the CDC CYBER 170, model
	171 is also included. This manual obsoletes all previous editions.
C	Revised to reflect NOS 1.3 and TAF 1.1 at PSR level 472, support only the time-sharing version of TAF,
(05-26-78)	and make typographical and technical corrections. This manual obsoletes all previous editions.
D	Revised to reflect NOS 1.3 at PSR level 485, and to make technical corrections.
(12-22-78)	
E	Revised to reflect NOS 1.4 and TAF 1.2 at PSR level 501 and to make technical and editorial
(08-10-79)	corrections. Featuresadded are CYBER Record Manager data manager support, COBOL version 5
	enhancements, requests (BWAITINP, TPSTAT, BEGIN, KPOINT, LOGIN, EXTRACT, and INSERT), TAF dump
	capability, LIBTASK enhancements, and changes to appendix B. This edition obsoletes all previous
	editions.
F	Revised to reflect NOS 1.4 at PSR level 530 and to make technical and editorial corrections. Support
(12-05-80)	of TAF/CDCS interface is included. This edition obsoletes all previous editions.
Publication No. 60453000	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

© 1976, 1977, 1978, 1979, 1980
by Control Data Corporation
All rights reserved
Printed in the United States of America

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected.

PAGE	REV
Front Cover	-
Inside Front Cover	F
Title Page	-
ii	F
iii/iv	F
v	F
vi	F
vii	F
viii	F
1-1	F
1-2	F
1-3	F
1-4	F
2-1	F
2-2	F
2-3	F
2-4	F
2-5	F
2-6	F
2-7	F
2-8	F
2-9	F
2-10	F
2-11	F
2-12	F
2-13	F
2-14	F
2-15	F
2-16	F
2-17	F
2-18	F
2-19	F
3-1	F
4-1	F
4-2	F
4-3	F
4-4	F
4-5	F
4-6	F
5-1	F
5-2	F
5-3	F
5-4	F
6-1	F
6-2	F
6-3	F
6-4	F
6-5	F
6-6	F
7-1	F
7-2	F
8-1	F
8-2	F
8-3	F
9-1	F
9-2	F
9-3	F

PAGE	REV
9-4	F
9-5	F
10-1	F
10-2	F
10-3	F
10-4	F
11-1	F
11-2	F
12-1	F
12-2	F
A-1	E
A-2	E
B-1	E
B-2	F
B-3	F
B-4	F
B-5	F
B-6	F
B-7	F
B-8	F
B-9	F
B-10	F
B-11	F
B-12	F
B-13	F
B-14	F
B-15	F
B-16	F
B-17	F
B-18	F
B-19	F
B-20	F
C-1	F
D-1	F
D-2	F
E-1	F
F-1	F
F-2	F
F-3	F
F-4	F
F-5	F
Index-1	F
Index-2	F
Index-3	F
Comment Sheet	F
Back Cover	-

PAGE	REV

PAGE	REV

PAGE	REV

PREFACE

This manual documents the executive portion of the CONTROL DATA® Transaction Facility (TAF), version 1.2 with time-sharing (multiplexer) interface (TAF/TS). The remaining portion, the data manager supplied with TAF/TS, is documented in the TAF/TS Data Manager Reference Manual. Three other data managers, CDC® CYBER Record Manager, Total, and CYBER Database Control Systems are also provided.

TAF runs as a subsystem of the NOS operating system and can be used with COMPASS, COBOL 5, and FORTRAN.

Another version of TAF exists which has a Network Access Method (NAM) interface. The NAM interface uses 2550 communications processors instead of multiplexers. Other manuals are available to describe the NAM version of TAF. However, none of the TAF/TS manuals listed under Related Publications contain this information.

AUDIENCE DEFINITION

This manual is for systems analysts and applications programmers. A systems analyst is a programmer who knows COMPASS and is concerned with internal design and functions. An applications programmer is a programmer writing tasks in FORTRAN or COBOL. New users of TAF should consult the User's Guide in the following list as a first source of information. It is an introductory text for inexperienced users. The guide also contains sections on recovery design and programming techniques.

The user must be familiar with basic NOS operations such as running a compilation job. The NOS Batch User's Guide acquaints the new user with the fundamentals of NOS. The NOS Reference Manual, Volume 1 is for the high-level language applications programmer who is already familiar with NOS. The NOS Reference Manual, Volume 2 is for the COMPASS applications programmer.

TERMINOLOGY AND CONVENTIONS

References in this manual to the transaction subsystem imply the whole of TAF/TS. References to the transaction executive imply the controlling portion of TAF/TS; that is, the portion described in this manual.

References to FORTRAN imply either FORTRAN Extended version 4 or version 5. References to COBOL imply either COBOL version 4 or 5 (the UC1 parameter must be included on the COBOL5 control statement).

The term alphanumeric refers to any combination of the letters A through Z and the numbers 0 through 7; special characters are not included.

Conventions for word formats are as follows:

- Cross-hatching indicates a field is not used by or is not applicable to TAF.
- Reserved fields might contain information useful to TAF internal processing. The user should not expect them to contain any particular values.
- Fields labeled with mnemonics indicate that a specific parameter must be inserted (generally described after the word format).
- Fields with numeric identifiers indicate the actual value that is used or returned for a particular request.

Extended memory for the CYBER 170 Model 176 is large central memory extended (LCME). Extended memory for all other NOS computer systems is extended core storage (ECS) or extended semiconductor memory (ESM).

In this manual, the acronym ECS refers to all forms of extended memory unless otherwise noted.

Programming information for the various forms of extended memory can be found in the COMPASS Reference Manual and in the appropriate computer system hardware reference manual.

RELATED PUBLICATIONS

The following manuals contain additional information for the user.

Control Data Publication	Publication Number
COBOL Version 4 Reference Manual	60496800
COBOL Version 5 Reference Manual	60497100
Common Memory Manager Version 1 Reference Manual	60499200
COMPASS Version 3 Reference Manual	60492600
CYBER Database Control System Version 2 Reference Manual	60481800
CYBER Loader Version 1 Reference Manual	60429800
FORTRAN Extended Version 4 Reference Manual	60497800
FORTRAN Version 5 Reference Manual	60481300
NOS Version 1 Batch User's Guide	60436300
NOS Version 1 Installation Handbook	60435700
NOS Version 1 Manual Abstracts	84000420
NOS Version 1 Operator's Guide	60435600
NOS Version 1 Reference Manual, Volume 1	60435400

Control Data Publication	Publication Number
NOS Version 1 Reference Manual, Volume 2	60445300
NOS Version 1 System Maintenance Reference Manual	60455380
NOS Version 1 Time-Sharing User's Guide	60436400
NOS Version 1 Time-Sharing User's Reference Manual	60435500
TAF/TS Version 1 CYBER Record Manager Data Manager Record Manager	60456700
TAF/TS Version 1 Data Manager Reference Manual	60453100
TAF/TS Version 1 User's Guide	60436500
Total-CDC Version 1 Reference Manual	76070300

The NOS Manual Abstracts is a pocket-sized manual containing brief descriptions of the contents and intended audience of all NOS and NOS product manuals. The abstracts can be useful in determining which manuals are of greatest interest to a particular user.

Control Data also publishes a Software Publications Release History (publication number 60481000), of all software manuals and revision packets it has issued. This history lists the revision level of a particular manual that corresponds to the level of software installed at the site.

DISCLAIMER

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

1. INTRODUCTION 1-1

On-Line Transaction Processing Advantages 1-1
On-Line Transaction Processing Problems 1-1
Batch Interface 1-3
Tasks 1-3
Entering Transaction Subsystem 1-3
Transaction Processing 1-3
TAF Requests 1-3
Implementing an Application 1-3
Subcontrol Points 1-4
Multimainframe 1-4
Extended Core Storage (ECS) 1-4

2. TASK/TRANSACTION EXECUTIVE INTERFACE 2-1

Communication Block 2-1
Requests 2-1
- Data Manager Requests 2-3
- Journal File Requests 2-3
- Memory Dump Requests 2-3
- System Requests 2-3
- Memory Requests 2-3
- Task Scheduling Requests 2-3
  - CALLRTN Request 2-4
  - CALLTSK Request 2-5
  - NEWTRAN Request 2-6
  - TRANCHK Request 2-6
- Input/Output Requests 2-6
  - WAITINP Request 2-7
  - LOADCB Request 2-9
  - SEND Request 2-9
  - RELSCB Request 2-11
  - TSIM Request 2-11
  - TARO Request 2-12
  - IIO Request 2-14
  - BWAITINP Request 2-14
  - TPSTAT Request 2-14
  - BEGIN Request 2-14
- Task Control Requests 2-16
  - RECALAL Request 2-16
  - CEASE Request 2-16
  - ITL Request 2-16
  - LOGT Request 2-17
  - KPOINT Request 2-17
  - WAIT Request 2-18
- Task Utility Requests 2-18
  - LOGIN Request 2-18
  - EXTRACT Request 2-19
  - INSERT Request 2-19

3. ACCESS AND USAGE OF CDCS BY TASKS 3-1

4. SYSTEM FILES 4-1

xxJ File 4-1
- TAF Data Manager xxJ File 4-1
- CRM Data Manager xxJ File 4-2
- TOTAL Data Manager xxJ File 4-3
- CDCS or No Data Manager xxJ File 4-3
- Creating xxJ File 4-4

Journal Files 4-4
- System Journal File 4-4
- Data Base Journal Files 4-4
- Journal File Entry Header 4-4
- JOURNL Request 4-4

TAF Configuration File 4-6

5. BATCH/TRANSACTION SUBSYSTEM INTERACTION 5-1

BTRAN Request 5-1
SUBMT Request 5-3

6. MEMORY DUMPS 6-1

Task Dump 6-1
- DSDUMP Request 6-1
- CMDUMP Request 6-4
- K.DSDUMP Command 6-4
- KTSDMP Utility 6-5
- Memory Dump Formats 6-5

TAF Dump 6-5
- K.DUMP Command 6-5
- K.DUMPLIM Command 6-5

7. RECOVERY/OPERATION INTERFACE 7-7

Recovery 7-1
Operator Interface 7-2

8. NETWORK FILES 8-1

Terminal Network Description File 8-1
VALNET Validation 8-2
Network File Organization 8-2

9. TASK LIBRARY 9-1

LIBTASK Utility 9-1
Task Residency and Type 9-1
LIBTASK Control Statement 9-2
Input Directives 9-3
Scheduling Priority 9-4
Creating a New Task Library 9-4
Updating an Existing Task Library 9-5
Deleting a Task from a Task Library 9-5
Ignoring a Task in the Replacement File 9-5

10. REQUIRED SYSTEM TASKS 10-1

Initial Task (ITASK) 10-1
TIMCNT Macro 10-1
TRAN Macro 10-1
STRAN Macro 10-2
ITASK Description 10-2
TAF-Originated Transactions 10-2
KDIS 10-3
LOG-OUT TASK (LOGT) 10-3
Message Abort (MSABT) 10-3
OFFTASK 10-4
SYSMSG 10-4
Execute Named Task (XTASK) 10-4

11. TERMINAL INTERFACE 11-1

Modes of Operation 11-1
Switch to Transaction Subsystem 11-1
Initiate Transaction Processing 11-1
Switch to Time-Sharing Subsystem 11-1
Forced Switch to Time-Sharing Subsystem 11-1
Message Size Limits 11-2
Special Characters 11-2
Time-Sharing Command Subset 11-2

12. BINARY TERMINAL I/O 12-1

Selection of Binary I/O 12-1
Set Binary Input Mode (0006) 12-1
Initiate Binary Output (0007) 12-1
Terminal Control 12-1
Identification of Binary Terminals 12-1
Code Provided for Binary I/O 12-1
User-Written Code for Binary I/O 12-2

APPENDIXES

A. CHARACTER SETS A-1
B. ERROR MESSAGES B-1
C. SAMPLE DECK STRUCTURES C-1
D. FILE DESCRIPTIONS AND DEFINITIONS D-1
E. DETECTION OF POTENTIALLY BLOCKED TASKS E-1
F. INSTALLATION OVERVIEW F-1

INDEX

FIGURES

1-1 TAF Control Point for CRM, TOTAL, and TAF Data Manager 1-2
1-2 TAF Control Point for CDCS Data Base Manager 1-2
2-1 Communication Block 2-2
2-2 CALLRTN 2-4
2-3 CALLRTN (Chain) 2-4
2-4 CALLRTN (Multiple Chains) 2-5
2-5 CALLTSK With CEASE 2-6
2-6 CALLTSK Without CEASE 2-6
2-7 NEWTRAN Parameter Table 2-7
2-8 TRANCHK Parameter Table 2-7
2-9 LOADCB Parameter Table 2-10
2-10 SEND Parameter Table 2-11
2-11 TSIM Parameter Table 2-13
2-12 TARO Parameter Table 2-15
2-13 IIO Parameter Table 2-15
2-14 TPSTAT Return Values 2-15
2-15 ITL Parameter Table 2-17
4-1 Journal File (Buffered and Nonbuffered Structure) 4-2
4-2 Journal File Entry Header 4-5
4-3 JOURNL Parameter Format 4-5
5-1 BTRAN Header Word 5-1
5-2 BTRAN Request 5-2
5-3 SUBMT Parameter Table 5-3
5-4 SUBMT Request 5-4
6-1 DSDUMP Parameter Table 6-3
6-2 CMDUMP Parameter Table 6-4
6-3 KTSDMP Format 6-6
8-1 Communication Between Terminal Status Table and Communication Block 8-3
10-1 Format of TTOT Table Entry 10-2
10-2 Format of TRANT Table Entry 10-2
10-3 Format of TRANS Table Entry 10-3
10-4 Communication Block for System Origin Transactions 10-4

TAF is a product that controls transaction processing. Transaction processing consists of taking an existing collection of information, called a data base, and correcting old data or adding new data to create an up-to-date data base. Such a correction or addition is called a transaction.

In a transaction processing system, the data base must be efficiently structured and easily accessible to the user. A data manager provides these features.

Control Data offers the following data managers for use with TAF.

- The CYBER Database Control System (CDCS)
- The CYBER Record Manager (CRM) (TAF supports only those features documented in the TAF CRM Data Manager Reference Manual).
- The Total data manager
- The TAF data manager

All four data managers can operate concurrently. However, a single transaction can use only:

- CDCS and one other data manager,

  or

- CDCS alone,

  or

- One other data manager alone.

The main function of TAF is on-line transaction processing (direct operator interface with the data base through a terminal). Advantages and problems are associated with an on-line system.

ON-LINE TRANSACTION PROCESSING ADVANTAGES

Some advantages associated with on-line transaction processing are:

- The operator submitting the data through TAF can make immediate data corrections.
- The operator can verify that operations have completed successfully. TAF allows the operator to view additions or corrections to the data base to verify their correctness.
- The data base is updated in seconds. TAF provides immediate updating capability which allows other transactions to make use of the updated data base.

The most important feature of TAF is its efficient system resource utilization. TAF's transaction processing differs from other modes of operation in that a task only contains application code. Terminal communications are handled by one terminal communications interface. Data base communications are handled by one copy of a data base manager. When tasks need to be loaded, they are simply copied from a task library to a subcontrol point; none of the normal overhead of loading is incurred.

The TAF control point contains application tasks as subcontrol points. When a task completes, one of the following occurs:

- It may be immediately reused by other waiting transactions (no reloading is necessary).
- Another task may be loaded in its place.
- Other active tasks may be moved up in memory and the remaining memory returned to the system for use by non-TAF applications.

TAF will expand its memory to a maximum value as needed.

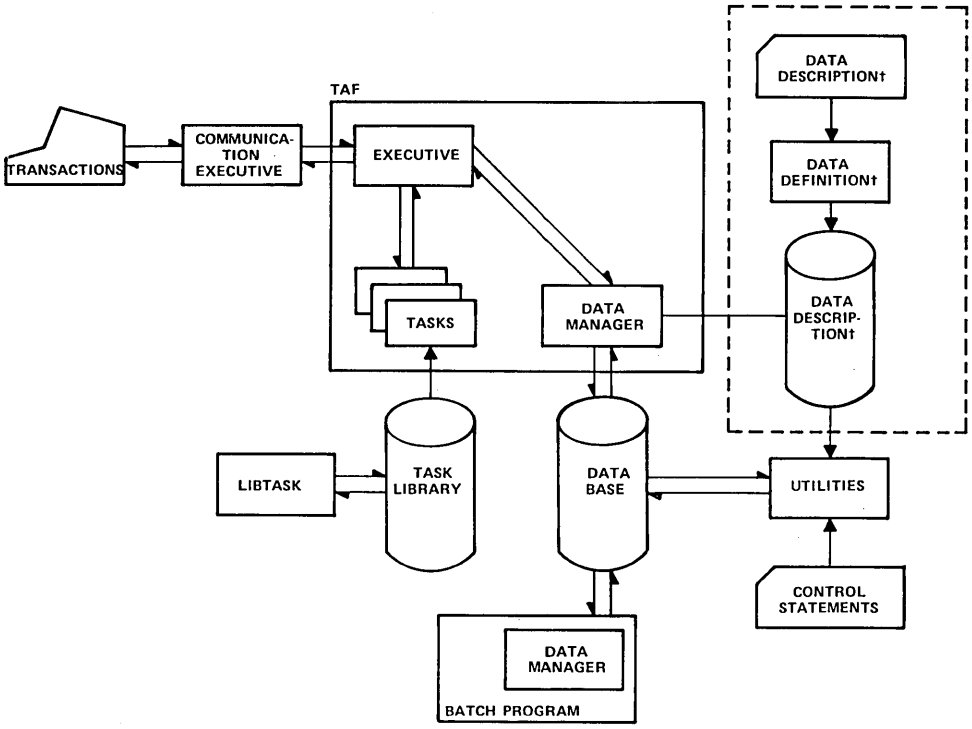
When there is no transaction activity, TAF size is reduced to a small amount of memory. When input is sensed, TAF rolls back in to service the request.

Figures 1-1 and 1-2 summarize the TAF control point and its operation.

ON-LINE TRANSACTION PROCESSING PROBLEMS

Some problems associated with on-line transaction processing and the methods that TAF uses to alleviate these problems are:

- It is difficult to schedule and control multiple jobs. TAF solves the problem of multiple job control scheduling by using subcontrol points. Like the operating system which controls and schedules jobs to control points (where they are processed), TAF controls and schedules transaction processing programs to portions of its control point called subcontrol points. Subcontrol points are protected from each other by CYBER hardware memory protection.
- The terminal interface is complicated. TAF minimizes terminal programming by using simple input/output (I/O) commands that are processed by the standard time-sharing terminal interface.



†COMPONENTS INSIDE DOTTED LINE NOT APPLICABLE TO CRM.

Figure 1-1.TAF Control Point for CRM, TOTAL, and TAF Data Manager.

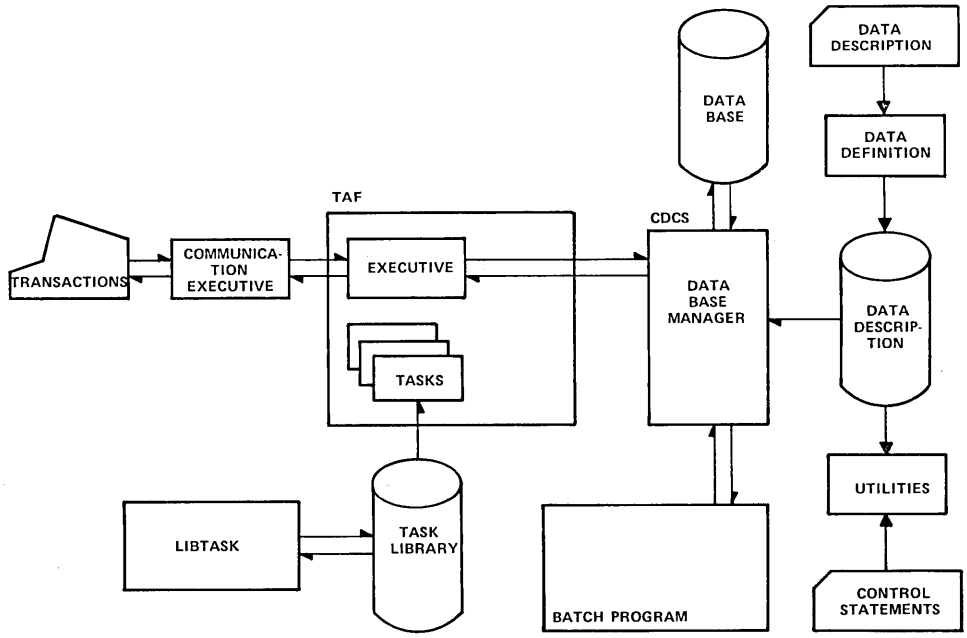


Figure 1-2. TAF Control Point for CDCS Data Base Manager.

BATCH INTERFACE

A batch interface with TAF is also available. The batch interface can generate large reports without tying up a terminal until the listing has been printed. The report can be printed at a high-speed central site printer. Batch mode can also be used for testing and debugging. The batch interface is described in section 5.

TASKS

The applications programmer implements a transaction system by writing tasks. A task is simply a program with a TAF interface that performs a specific function. A single task or several tasks perform a sequence of events called a transaction. Tasks can read and update information on the user's data base. They can send messages to terminals and receive input. Tasks can also schedule other tasks to assist in the completion of transactions.

Tasks that run under TAF can be written in COBOL, FORTRAN, or COMPASS. A task differs from a normal batch or time-sharing program in that it contains calls to TAF commands and data manager commands and must have a common block (referred to as the communication block) defined. The communication block is needed to receive terminal input and pass data between tasks. (The communication block is described in section 2.)

Binary copies of each task are stored in the task library. The LIBTASK utility is used to place these binary copies on the task library. Section 9 describes the task library and LIBTASK.

ENTERING TRANSACTION SUBSYSTEM

After logging into the time-sharing system,† a terminal user must request the transaction subsystem. To do this, the user enters TRAN,terminal name (CR). The terminal name is specified in a file called the network file which defines the whole terminal configuration. The network file is described in section 8. When the system responds with READY, the user can proceed with transaction processing. Further details are given in section 11.

TRANSACTION PROCESSING

After entering the transaction subsystem, the user indicates the transaction to be performed by entering a code that is assigned by the system designers. Such a code could specify that a user is entering a loan payment; another code would be used to close out a loan, and so on. TAF receives this initial input and places it in a communication block. This communication block is the means by which the data is passed from a terminal to a task or from task to task.

When a transaction is completed, the communication block is released and can be used by other transactions.

After the communication block has been constructed, TAF schedules an initial task (ITASK) to interpret the code the user entered. Using this code, ITASK requests TAF to schedule the appropriate task to process the transaction. The task scheduled may also call other tasks to assist in the completion of the transaction.

TAF REQUESTS

Communication between terminals and tasks is accomplished by using TAF requests. Output can be sent to a transaction terminal from the task by the use of the SEND request. When this request is processed, control passes to TAF. TAF extracts the message from the task and passes it to the time-sharing system. The message is then transmitted to the transaction terminal, and control passes back to the task for additional processing.

When a task requires additional input from the terminal, the WAITINP request is used. The task performs a SEND request to send output to the terminal, prompting the terminal operator for the needed input. The WAITINP request is then processed. Since the terminal operator takes seconds to respond to the request, the task is rolled out and the subcontrol point and communication block are released. These resources can then be used to process other transactions while the terminal operator is responding. When TAF receives the requested input, it places it into a new communication block, and the task is rolled into an available subcontrol point. The new communication block is passed to the task, and the task continues processing.

These and other requests are further described in sections 2 and 5.

IMPLEMENTING AN APPLICATION

The following steps specify the procedure for implementing an application.

1. Review this manual.
2. Design the application including:
   - Selection of the data manager (CDCS and/or CRM, TOTAL, or TAF) to allow the correct choice of file types, file structure, and file interaction.
   - Selection of recovery and backup techniques, including journalizing of files.

   Refer to the CYBER Database Control System Reference Manual, the CRM Data Manager Reference Manual, the TOTAL Reference Manual, and the TAF/TS Data Manager Reference Manual for more information.
3. Describe and install the data base using the data manager. Refer to the appropriate data manager reference manual for this procedure.
4. Test load the data base using the data manager commands (usually done in batch mode).

†Refer to the Time-Sharing User's Reference Manual (publication number in preface) for a description of the log-in procedure.

5. Using the application design, write the application tasks to process transaction input.

6. Modify ITASK to reflect the actual application. Tasks MSABT, LOGT, and SYSMSG may also require modification. (Refer to section 10.)

7. Build a task library. (Refer to section 9.)

8. Build an xxJ file. (Refer to section 4.)

9. Build a network file. (Refer to section 8.)

10. Build a transaction configuration, TCF file. (Refer to section 4.)

11. Test the system.

SUBCONTROL POINTS

The following discussion is intended for analysts and may be omitted by the applications programmer.

Tasks reside at subcontrol points within the field length of the transaction executive, which resides at control point 2. Each task runs with its own exchange package, reference address (relative to the transaction executive), and field length (a subset of the field length of transaction executive). A task, therefore, can reside in any contiguous segment within the transaction executive field length. Memory is allocated in 100_8 word blocks located on even 100_8 word boundaries (for example, 32100_8 to 32200_8). A subcontrol point consists of one 100_8 word block of memory for system use, followed by sufficient blocks to contain an entire task.

The subcontrol point feature allows the transaction executive to control each task. Some of the advantages associated with the subcontrol points are:

- Isolating one subcontrol point from other subcontrol points and the transaction executive. This means that no application program can destroy the transaction executive or circumvent system security.
- Blocking RA+1 requests from a subcontrol point. No requests are allowed directly from a subcontrol point to NOS. Any such requests are intercepted by the system monitors, which return control to the transaction executive. Thus, the only RA+1 requests a task can legally issue are those processed by the transaction executive. This includes requests to the data manager.

The transaction subsystem allows a maximum of 31 subcontrol points. An installation parameter sets the number of control points that the transaction executive initializes. When the transaction executive is loaded, the operator can select a number of subcontrol points other than this default value. The number of subcontrol points must not be less than 2 nor greater than 31. Once the transaction subsystem is initialized, no change in the number of subcontrol points is allowed. The optimum number of subcontrol points is selected by the site.

Each subcontrol point requires eight words of table space. No space, other than a table entry, is allocated for a subcontrol point unless it is active. Each subcontrol point has 111_8 words reserved, beginning at RA, similar to those used for batch control points.

MULTIMAINFRAME

Under a multimainframe configuration, the transaction subsystem can be run on either or both mainframes. However, a data base and JOUR0 file (section 4) cannot be shared by the transaction subsystems. Each subsystem must have a unique data base.

EXTENDED CORE STORAGE (ECS)

TAF does not allow the use of the user access to ECS feature. However, the TAF procedure file can declare data files and libraries as ECS-resident.

A task is isolated from the data manager and the remainder of the system by the transaction executive. The isolation guarantees that errors in a task do not propagate beyond the task. The following types of requests constitute the interface between the task and the transaction executive.

- Data manager requests
- Journal file requests
- Memory dump requests
- System requests
- Task scheduling requests
- Input/output requests†
- Task control requests
- Task utility requests

Internally, a task performs a request by loading address RA+1 of its subcontrol point. This activity is not visible to the application programmer who uses the macros or calls given in this section. The transaction executive uses the request parameters to determine what processing is required by the task. The address field and any fields in a parameter list that reference an address contained within the task are verified to ensure that the address referenced is within the task field length.

Tasks communicate with each other and accumulate input by using the communication block. The communication block is discussed next, followed by the types of requests.

COMMUNICATION BLOCK

Communication between a chain of tasks is provided by the communication block. The initial input or the input from a terminal as a result of the WAITINP (refer to Input/Output Requests later in this section) request is also stored in the communication block. The communication block can be used to pass parameters or data from task to task. The format of the communication block is shown in figure 2-1. The length of terminal input is not restricted by the length of the communication block. The LOADCB request is available to read extra input that overflows the communication block (refer to the description of i in figure 2-1). If the communication block is passed between tasks in a transaction and overflow data is present, other tasks in the transaction can access this data with a LOADCB request.

If COBOL is used, the communication block must be described in the COMMON-STORAGE SECTION. Shown here is an example of such a description:

```
        .
        .
        .
DATA DIVISION.
COMMON-STORAGE SECTION.
01 COMMUNICATION-BLOCK.
    03 CB-HEADER-AREA-WORD-ONE.
        05  CB-DATA-BASE-NAME                  PIC X(2).
        05  CB-USER-TST-AREA          COMP-4   PIC 9(6).
        05  CB-TRANS-SEQ-NUMBER       COMP-4   PIC 9(6).
    03 CB-HEADER-AREA-WORD-TWO.
        05  CB-USERNAM                         PIC X(7).
        05  CB-INPUT-STATUS-BYTE      COMP-4   PIC 9.
        05  CB-MESSAGE-WORD-COUNT     COMP-4   PIC 99.
    03 CB-MESSAGE-AREA.                        PIC X(660).
    03 CB-TRAILER-AREA.
        05  FILLER                             PIC X(4).
        05  CB-PACKED-DATE.
            07  CB-YEAR               COMP-4   PIC 9.
            07  CB-MONTH              COMP-4   PIC 9.
            07  CB-DAY                COMP-4   PIC 9.
      05    CB-PACKED-TIME.
            07  CB-HOUR               COMP-4   PIC 9.
            07  CB-MINUTE             COMP-4   PIC 9.
            07  CB-SECOND             COMP-4   PIC 9.
WORKING-STORAGE SECTION.
        .
        .
        .
```

This configuration will be referred to in this manual as the Standard COBOL Communication Block because of the correspondence between it and the fields shown and described in figure 2-1. However, applications programmers may find it convenient to use a more abbreviated communication block when a particular case allows.

COMP-4 is specified for data areas intended for numerical information. Refer to the COBOL 5 Reference Manual for information on the relationship between the number of digits specified in the picture clause for COMP-4 data and the number of bits reserved.

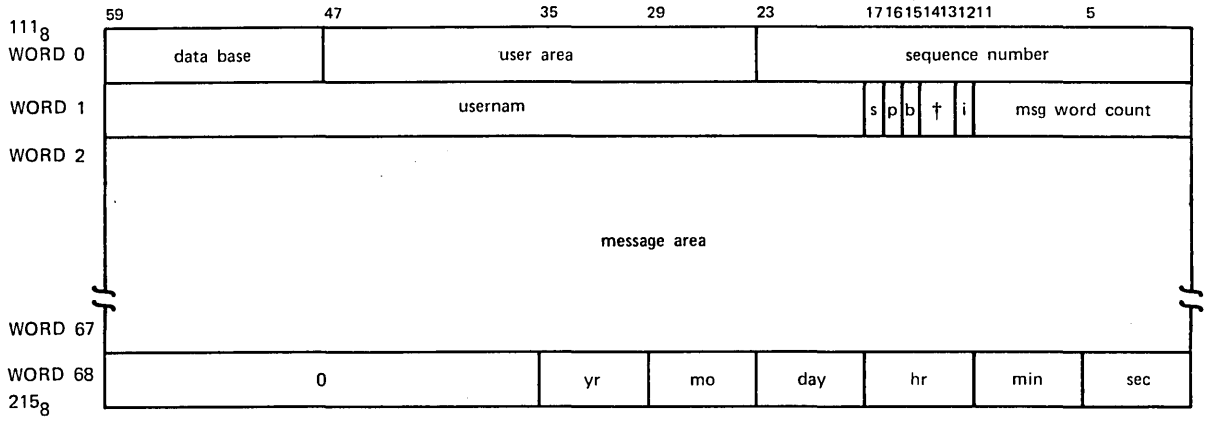
If FORTRAN is used, the communication block must be in the first block of labeled common.

If more data is input than can be contained in the communication block, the LOADCB request may be used to obtain the input.

REQUESTS

The following paragraphs describe the various types of requests a task can make. They describe task scheduling, input/output, task control, and task utility requests and show the formats for making these requests from tasks written in FORTRAN, COBOL, and COMPASS.

†The standard CRM-supported, compiler-generated I/O calls from FORTRAN and COBOL are not allowed. The COBOL SORT verb is not supported.



data base	Data base the originating terminal is validated to use.
user area	Image of originating terminal user area from the terminal status table (refer to section 8). Contains the recovery flag. (Refer to the TARO and TSIM requests for more information.)
sequence number	Assigned by the transaction executive to each newly initiated transaction. This number is generated by incrementing a counter by 1 and identifies the transaction, or task chain, throughout its life in the subsystem.
terminal	One- to seven-character terminal name, left-justified with binary zero fill from the terminal status table.
s††	Bit 17 of word 1 is set to 1 if this communication block was created by a TAF-originated transaction. Word 2 of the communication block contains additional information (system identified conditions which may be processed by TAF-originated tasks). Refer to section 10 for information concerning required system tasks.
p††	Bit 16 of word 1 is set to 1 if this communication contains data on which a parity error occurred during terminal input.
b††	Bit 15 of word 1 is set to 1 if this communication block was created by a batch transaction (BTRAN request).
i††	Bit 12 of word 1 is set to 1 if additional input to be loaded by means of the LOADCB request exists.
msg word count	Length of the input message in words in the communication block proper.
message area	A message from a terminal, as a result of initial terminal input, WAITINP, or BWAITINP requests, begins at word 2 and is terminated by 12 bits of zero in positions 11-0 of the last word of the message. All remaining message area words contain binary zero. The maximum length of a terminal message is 57 words. This area may also contain data from a BTRAN request. The maximum length of a BTRAN message is 62 words. In all cases, the length of the message is in the msg word count field. The information contained in this area is available to the user, and its contents is passed to any communication block generated by the processing of a CALLTSK or CALLRTN request. If it is desired to retain the terminal input, the user may only write in words msg word count+2 through 67. If there is no need to retain the terminal input, the entire area can be used.
yr, mo, day, hr, min, sec	Data and time in packed binary format from the operating system PDATE macro.

Figure 2-1. Communication Block

†Reserved.

††The main purpose of these fields is to notify the system task ITASK of certain conditions. Under COBOL, the EXTRACT routine allows examination of these fields. Also, a COBOL data type, computational-4, can be used to access these fields more easily; computational-4 data types are allowed only on COBOL version 5.2 and above. ITASK is described in section 10, and EXTRACT is described later in this section.

DATA MANAGER REQUESTS

A task can make data manager requests to one of the data managers (CDCS, CRM, TOTAL, or TAF) supported by the transaction subsystem. (Refer to the preface for the publications that document these data managers and the requests used.)

JOURNAL FILE REQUESTS

A task can supplement automatic journaling by writing data on a journal file. Journal files and the journal requests are discussed in section 4.

MEMORY DUMP REQUESTS

These requests are discussed in section 6.

SYSTEM REQUESTS

System requests from transaction tasks are limited to the following:

COMPASS Macros

ABORT

DATE x

ENDRUN

JDATE x

MEMORY (as described below)

MESSAGE x (to console B display, line 1 only)

PDATE x

RTIME x

TIME x

FORTRAN Functions and Subroutine Calls

DATE(x) or CALL DATE(x)

JDATE(x) or CALL JDATE(x)

TIME(x) or CALL TIME(x)

COBOL Statements

ACCEPT x FROM DATE.

ACCEPT x FROM TIME.

ACCEPT item FROM TTY.

DISPLAY item UPON TTY.

These requests are described in detail in volume 2 of the NOS Reference Manual, FORTRAN Extended 4 Reference Manual, and the COBOL 5 Reference Manual. (Refer to the preface for publication numbers.)

MEMORY REQUESTS (COMPASS only)

Tasks are allowed to interrogate the status of and change their CM field length (FL) via the RA+1 requests MEM and RFL. The format of these calls is documented for the MEMORY macro in the NOS Reference Manual, volume 2. The use of these requests in TAF is restricted as follows:

- Memory requests should be issued with recall (r parameter); otherwise control may be returned before the request is processed.
- There is an upper limit beyond which a task is not allowed to increase its CM space; this limit is referred to as the maximum task field length (MFL). The MFL for each task is equal to the task FL plus the value specified by the EF LIBTASK directive for the task (refer to section 9). If a task attempts to exceed this value, it may be aborted (refer to documentation of the MEMORY macro NA parameter in the NOS Reference Manual, volume 2).
- Field length cannot be decreased to less than the last word address of the default communication block transfer address rounded up to the nearest 100_8.
- A non-CM resident task making a memory request may be rolled out of CM if the request cannot be satisfied initially; therefore, a task must have no outstanding (incomplete) data manager requests.
- Since a task making a memory request may be rolled out, it is usually desirable to free locked data manager resources prior to the request.

Tasks will also be allowed to use the Common Memory Manager (CMM) with the following restrictions:

- The processing performed by CMM for the task must not result in CMM using the Fast Dynamic Loader or the task will be aborted.
- The above restrictions on use of MEM and RFL requests apply.

Common Memory Manager requests are documented in detail in the Common Memory Manager Version 1 Reference Manual.

TASK SCHEDULING REQUESTS

A task, including ITASK (refer to section 10), can call, in order, a list of other tasks. This process is referred to as task scheduling. The following four requests can be used to schedule tasks.

- CALLRTN

  Calls up to five tasks. Upon completion of the called chain of tasks, returns control to the calling task.

- CALLTSK

  Calls up to five tasks. The task making the call either ceases or continues to run, allowing the new task chain to proceed independently.

- NEWTRAN

  Initiates up to five new tasks (COMPASS only).

- TRANCHK

  Checks to see if a specific task is active (COMPASS only).

NOTE

If a list has either no tasks or more than five tasks, TAF aborts the task and issues the following error message on the screen.

TASK REQUEST ARGUMENT ERROR

CALLRTN Request

The CALLRTN request is used to request that the transaction executive schedule a task or series of tasks, with control returning to the calling task upon completion of the called tasks. A CALLRTN request cannot be made while data manager requests are outstanding. The current communication block is passed to the called tasks and returned to the caller when the caller is reinitiated. Upon return, the caller begins execution immediately after the CALLRTN request. Only the information in the communication block may have changed. Return from the called tasks to the caller takes place automatically after the last called task ceases. A task waiting for return may be temporarily rolled from memory.

The format is:

COBOL

ENTER CALLRTN USING $task_1$ $task_2$...$task_n$.

$task_i$ Parameter whose value is a one- to seven-character alphanumeric task name. The tasks are scheduled by CALLRTN in the order specified. From one to five tasks can be specified.

FORTRAN

CALL CALLRTN($task_1$,$task_2$,...,$task_n$)

$task_i$ Parameter whose value is a one- to seven-character alphanumeric task name, left-justified with either blank or binary zero fill to a word boundary. The tasks are scheduled by CALLRTN in the order specified. From one to five tasks can be specified.

COMPASS

CALLRTN addr

addr Address of a list of from one to five task names. Each task name must be left-justified with binary zero fill, to a word boundary. The list is terminated by a word of binary zeros. Tasks are scheduled by CALLRTN in the order they appear on the list.

NOTE

If $task_i$, specified on a CALLRTN request, makes a CALLRTN request, $task_{i+1}$ through $task_n$, specified on the original CALLRTN request, are not scheduled (refer to figure 2-4).

The following diagrams illustrate CALLRTN processing.

In figure 2-2, TASK A, at subcontrol point X, executes a CALLRTN to TASK B. TAF will first load a copy of TASK B to an available subcontrol point (unless a serially reusable copy of TASK B is already at a subcontrol point or CM resident). TAF will then copy the contents of TASK A's communications block (C.B.) to TASK B's communication block area. TASK B begins execution. When TASK B has completed execution, TAF will copy the communications block (which was probably modified by TASK B) back to TASK A and allow TASK A to continue execution at the instruction immediately following the CALLRTN command.

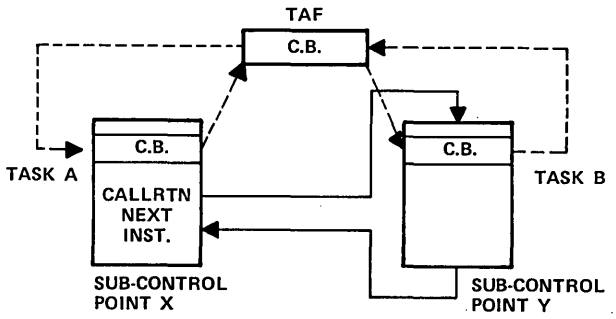


Figure 2-2. CALLRTN

The CALLRTN command allows a list (chain) of up to five tasks to be called. Figure 2-3 shows TASK A calling a chain of three tasks. As each task in the chain is completed, TAF will automatically pass the communications block to the next task in the chain and allow that task to execute. When the last task in the chain has completed, TAF will pass the communications block back to the calling task and allow it to continue execution at the instruction immediately following the CALLRTN command.

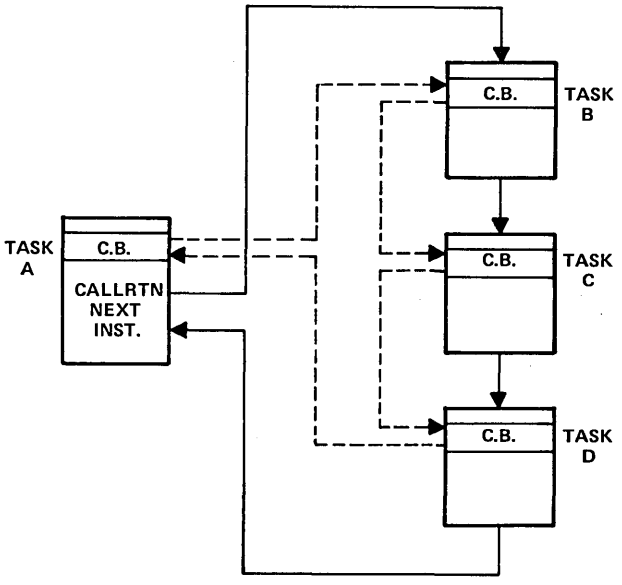


Figure 2-3. CALLRTN (Chain)

Figure 2-4 illustrates the processing referred to in the note above. The majority of the data inputs are handled by the TASK B-C-D chain. When an exception is found in TASK C, exception processing is invoked in the TASK E-F chain. Each new CALLRTN command cancels the previous one (TASK D is not executed), but TAF will always bring control back to the instruction immediately following the original CALLRTN command (TASK A).

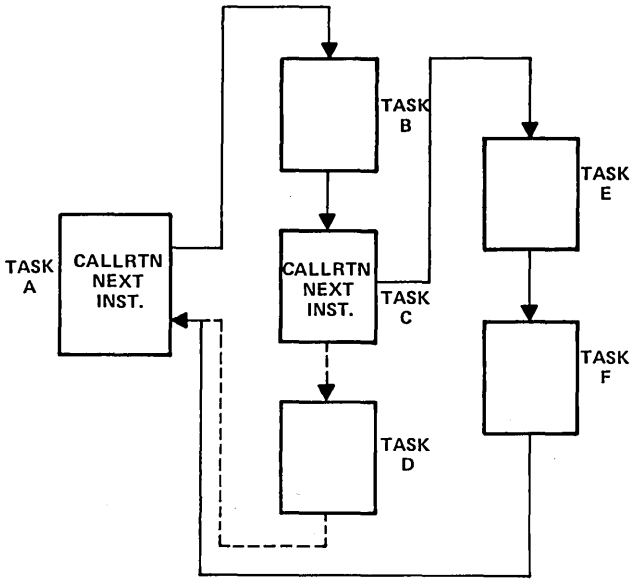


Figure 2-4. CALLRTN (Multiple Chains)

CALLTSK Request

The CALLTSK request is used to request that the transaction executive schedule a task or series of tasks (task chain). Included in this request is a parameter option which enables task branching or a CEASE function to be performed after the call is honored. If the task is not to cease on this call, a branching of the chain initiates a new chain which is independent of the original chain. The new chain is assigned a new sequence number for unique identification within the transaction subsystem (figure 2-1). The number of branches allowed by a transaction is controlled by an assembly constant MAXBW.

The format is:

COBOL

ENTER CALLTSK USING task code $task_1$...$task_n$.

task — Parameter whose value is a one- to seven-character alphanumeric task name. This task is the next task to call in the present task chain (same sequence number) if code is set to zero; otherwise, it is the first task of a branch chain (the called task gets a different sequence number and proceeds independently of the calling task).

code — Computational-1 item whose value is nonzero if the calling task is to continue after the request, initiating an independent task chain. If the value of this parameter is 0, the calling task ceases after this request. This parameter must be specified.

$task_i$ — Parameter whose value is a one- to seven-character alphanumeric task name. From zero to four tasks can be specified. These tasks are scheduled after the one specified in the task parameter.

FORTRAN

CALL CALLTSK(task,code,$task_1$,...,$task_n$)

task — Parameter whose value is a one- to seven-character alphanumeric task name, left-justified with either blank or binary zero fill to a word boundary. This task is the next task to call in the present task chain (same sequence number) if code is set to zero; otherwise, it is the first task of a branch chain (the called task gets a different sequence number and proceeds independently of the calling task).

code — Integer whose value is nonzero if the calling task is to continue after the request, initiating an independent task chain. If the value of this parameter is 0, the calling task ceases after this request. If not specified, the default is 0.

$task_i$ — Parameter whose value is a one- to seven-character alphanumeric task name, left-justified with either blank or binary zero fill to a word boundary. From zero to four tasks can be specified. These tasks are scheduled after the one specified in the task parameter.

COMPASS

CALLTSK addr,ceas

addr — Address of a list of from one to five task names. Each task name must be left-justified with binary zero fill to a word boundary. The list is terminated by a word of binary zeros. CALLTSK schedules the tasks in the order they appear on the list.

ceas — If specified, the calling task ceases after this request. If not specified, the calling task continues after the request. In this case the CALLTSK initiates an independent task chain.

The following diagrams illustrate CALLTSK processing.

Figure 2-5 shows TASK A making a CALLTSK with CEASE to TASK B. TAF will load a copy of TASK B to an available subcontrol point (this step is not necessary if a serially reusable copy of TASK B is already at a subcontrol point or CM resident). The communications block will be copied from TASK A to TASK B. TASK A's subcontrol point is released for use by other transactions, and TASK B is allowed to begin execution.

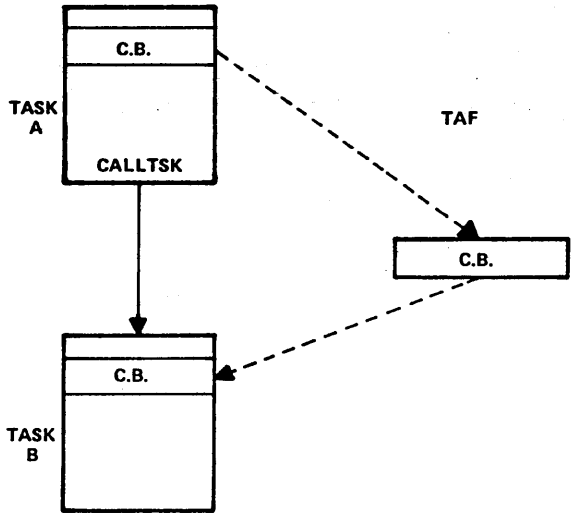


Figure 2-5. CALLTSK With CEASE

The CALLTSK without CEASE command allows a new asynchronous chain to be started. In figure 2-6, TASK A uses this command to call TASK B. TAF will copy the contents of TASK A's communications block to a new communication block. A copy of TASK B will be loaded to an available subcontrol point (unless a serially reusable copy of TASK B is already at a subcontrol point or CM resident). The new communications block will be copied to TASK B. TASK A is allowed to continue execution and TASK B is allowed to begin execution.

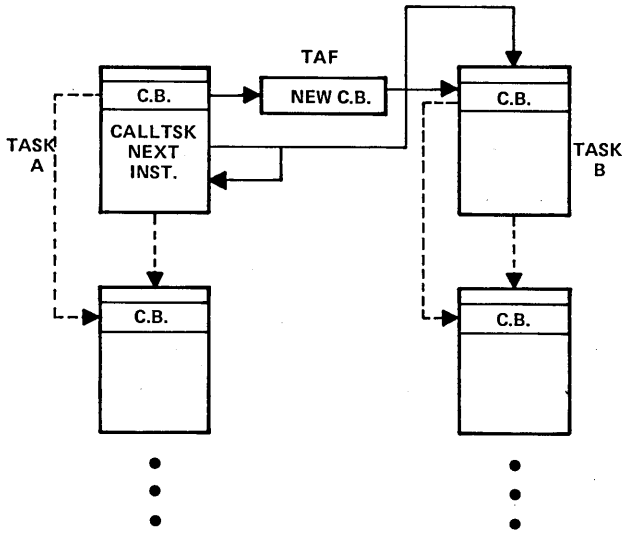


Figure 2-6. CALLTSK Without CEASE

NEWTRAN Request

The NEWTRAN request initiates a new transaction chain with a unique sequence number and with the data base, user area, and user name set to 0 in the communication block. The NEWTRAN request is designed for system use, primarily by ITASK, and is provided only for COMPASS.

The format is:

COMPASS

```
NEWTRAN  addr
```

addr	Address of a parameter table in the form shown in figure 2-7.

TRANCHK Request

The TRANCHK request determines the status of a transaction. This request is provided only for COMPASS.

The format is:

COMPASS

```
TRANCHK  addr
```

addr	Address of a parameter table in the form shown in figure 2-8.

INPUT/OUTPUT REQUESTS

The following requests are associated with input/output.

- WAITINP

  Places terminal input data in the communication block. Equivalent to a read request.

- LOADCB

  Reads data that overflowed the communication block.

- SEND

  Sends data to terminal. Equivalent to a write request.

- RELSCB

  Releases memory used for extra communication blocks.

- TSIM

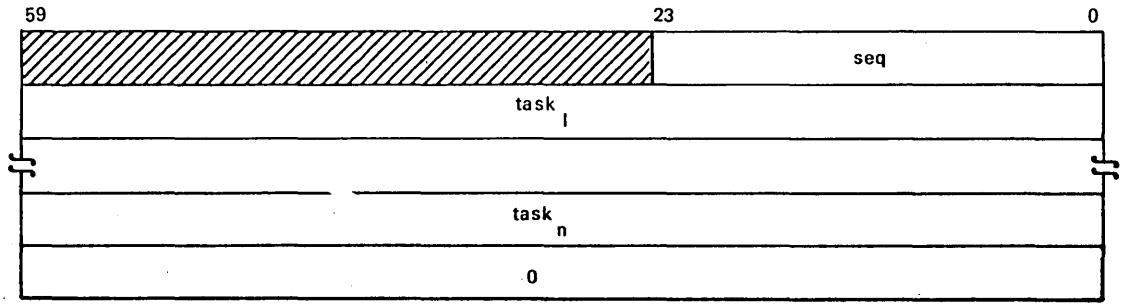
  Reads terminal status table.

- TARO

  Writes into terminal status table.

- IIO

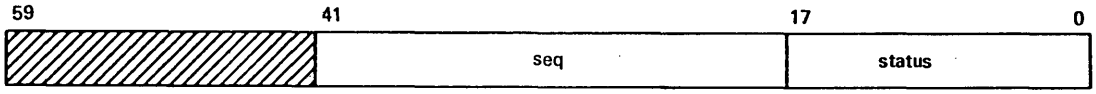
  Limits total number of requests (RA+1 calls) a task can make.



seq — The transaction executive returns a new 24-bit binary transaction sequence number in this field.

$task_i$ — One- to seven-character task name, let-justified with binary zero fill. From one to five tasks can be specified; the list is terminated by a word of binary zeros. NEWTRAN schedules the tassks in the order they appear on the list.

Figure 2-7. NEWTRAN Parameter Table



seq — Binary transaction sequence number which identifies the transaction (task or task chain).

status — The system returns one of the following as status in this field.

1 The task chain is still active.

0 The task chain is inactive.

Figure 2-8. TRANCHK Parameter Table

- BWAITINP

  Specifies that terminal input data is to be placed in a location in the task field length that is not reserved for the communication block.

- TPSTAT

  Returns a value which indicates whether the active terminal subsystem is TS or NAM.

- BEGIN

  Specifies the area where the terminal transmission is to be placed in the task field length. It does not return the communication block to the first word address of the task unless the task requests.

WAITINP Request

The WAITINP request (wait for terminal input) reads data from a terminal. Internally, this request informs the transaction executive that the next input received from a terminal is to be passed directly to the requesting task and is not the initiation of a new transaction. To a task, the data appears in the communication block. The data held in the communication block prior to the WAITINP is lost.

The first parameter is a data item or variable whose value the system sets to indicate the reception of terminal input or an error condition. An optional second parameter of the request specifies a time limit, after which processing of the transaction is restarted even if no input has been received from the terminal.

NOTE

At least one message (SEND) must be sent to the terminal before making a WAITINP request, and at least one SEND request from the original task chain must follow a message received following a WAITINP request. A task that includes a WAITINP request must not have been called by a CALLRTN request.

No data manager requests may be active when a task makes a WAITINP request. Thus, for transations using the TAF data manager, a RECALAL (refer to the RECALAL request later in this section) should precede a WAITINP if there is any chance that data manager requests may be outstanding. All tasks that make WAITINP requests must be entered in the task library with the scheduling queue limit set to 1. (Refer to section 9.)

WAITINP does not cause any data manager cease requests to be issued. Therefore, any resources locked or reserved by a transaction prior to a WAITINP, continue to be locked or reserved during and after the WAITINP.

When a WAITINP request is processed, all communication blocks are released and the task is rolled out. All data in the communication block except the transaction sequence number will be lost so the user should save any data to be preserved prior to the request. When input arrives from the terminal, a new communication block is generated, and the task is restarted. The new communication block will contain the transaction sequence number associated with the task prior to the WAITINP request. If the request does not return a normal status, the information in the communication block is meaningless.

The format is:

COBOL

ENTER WAITINP USING status time.

status — Computational-1 item which the system sets to indicate the status of the WAITINP request. Upon completion of the request, status has one of the following values.

- 0 — The request processed normally; terminal input appears in the normal input portion (begins in the third word) of the communication block.
- < 0 — Another task is waiting for input from the originating terminal. The WAITINP request must be reissued if input is desired.
- > 0 — The time delay elapsed before input was received from the terminal.

time — Computational-1 item which contains the time limit, in decimal seconds, to wait for input. If not specified, the default is 480 seconds. The maximum value is 2048 seconds.

FORTRAN

CALL WAITINP(status,time)

status — Integer variable which the system sets to indicate the status of the WAITINP request. Upon completion of the request, status has one of the following values.

- 0 — The request processed normally; terminal input appears in the normal input portion (begins in the third word) of the communication block.
- < 0 — Another task is waiting for input from the originating terminal. The WAITINP request must be reissued if input is desired.
- > 0 — The time delay elapsed before input was received from the terminal.

time — Integer time limit, in decimal seconds to wait for input. If not specified, the default is 480 seconds. The maximum value is 2048 seconds.

COMPASS

WAITINP time

time — Time limit, in decimal seconds to wait for input. If not specified, the default is 480 seconds. The maximum value is 2048 seconds.

Status of the WAITINP macro is determined by the following:

- Word 2 of the communication block contains 1 if the time delay elapsed before input was received from the terminal.
- The X6 register contains a negative value if another task is waiting for input from the originating terminal. The WAITINP request must be reissued if input is desired. (Word 2 of the communication block should be checked before X6 is checked.)
- The X6 register is set to 0 if the request processed normally; terminal input appears in the normal input portion (begins in the third word) of the communication block. (Word 2 of the communication block should be checked before X6 is checked.)

LOADCB Request

If the transaction input data exceeds the message area of the communication block, bit 12 of word 1 of the communication block (the i field) is set. The LOADCB request loads the remainder of the input message into a user-defined buffer. This buffer is a single contiguous block of storage for all additional communication blocks. No headers or trailers appear in this area.

The maximum input size is determined by an installation parameter.

The data that remains after the communication block message area fills is transferred to the user-defined buffer. If the LOADCB request is made with no data remaining (i field not set) or if a RELSCB request was made, no data is transferred and the status word is set to 0. If the remaining data consists of more words than will fit in the buffer, the buffer is filled, and the status word is set to a value greater than the number of words in the buffer. If the remaining data can be contained in the buffer, the remaining data is transferred to the buffer, and the number of words transferred is set in the status word.

The message data exceeding the communication block can be requested as many times during a transaction as desired. Each time, the overflow data is transmitted as specified previously.

The format is:

COBOL

ENTER LOADCB USING buff status r.

buff	A 01-level item identifying the user-defined buffer area.
status	Computational-1 item which the system sets to indicate the LOADCB request status. The significance of the value set has been described earlier in the LOADCB request description.
r	Computational-1 item. If a nonzero value is entered for this parameter, any message data that overflows the LOADCB buffer specified with the buff parameter is released (same as RELSCB).

FORTRAN

CALL LOADCB(buff,len,status,r)

buff	An array identifying the user-defined buffer area.
len	Integer length of the array specified with the buff parameter in words.
status	Variable which the system sets to indicate the LOADCB request status. The significance of the value set has been described earlier in the LOADCB request description. This value is also returned as the value of the function, if LOADCB is called as a FORTRAN function.
r	Integer variable. If a nonzero value is entered for this parameter, any message data that overflows the LOADCB buffer specified with the buff parameter is released (same as RELSCB).

As stated in the description of the stat parameter, the LOADCB request can be referenced as a FORTRAN function. The format of the function is LOADCB(buff,len,stat,r); the parameters are the same as those already described. The value of the function is the value of the stat parameter.

COMPASS

LOADCB addr

addr	Address of a parameter table in the format shown in figure 2-9.

The LOADCB macro status is returned in the X6 register. The significance of the value returned has been described earlier in the LOADCB request description.

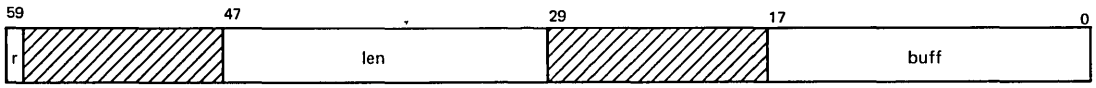
SEND Request

The SEND request sends a message to a terminal; that is, it performs a write operation. If the user does not specify the terminal to which the message should be sent, it is directed to the terminal that originated the transaction. At least one SEND request to the originating terminal must occur from the primary task chain for each message received. The user can also specify that the task be ended when the message is sent.

There is both a limit on the total output that one task can send to the time-sharing subsystem (MAXTO)† and a limit on the output one task can send before it is rolled out, pending the time-sharing subsystem's transfer of the output to the terminal or terminals (ROLTO).†

The system uses the following rollout procedure. A task can execute one SEND of any length (up to the maximum for one SEND) without being rolled out. The maximum for one SEND is determined by the MAXWS† installation parameter. However, if a task executes two or more SENDs, it may exceed the rollout send threshold (ROLTO). If the task message length exceeds this ROLTO value, it is rolled out. Any data manager resources held by the task, such as locked records and buffer space for any currently held records, remain assigned for the duration of the rollout. If reserving these resources for this period of time will impact the performance of the application, then tasks using the TAF data manager should issue a CLEAR†† request to return all data manager resources before it executes the SEND that causes the task to exceed ROLTO. After a rollout caused by exceeding the ROLTO limit, a task can again send as many characters as allowed by the ROLTO limit (up to MAXTO limit) before being rolled out again for exceeding the ROLTO limit.

†Refer to the NOS Installation Handbook (publication number is contained in preface).
††Refer to the TAF/TS Data Manager Reference Manual (publication number is contained in preface).



r	If bit 59 is set to 1, any message data that overflows the LOADCB buffer is released (same as RELSCB).
len	Length of the LOADCB buffer in words.
buff	First word address of the buffer area.

Figure 2-9. LOADCB Parameter Table

The format is:

COBOL

ENTER "SEND" USING mesaddr terminal cease-flag output-flag block status length.

mesaddr	Name of a 01- or lower-level item containing the message to be sent. 77-level items cannot be used; otherwise, truncation of the message occurs.
terminal	A 01 level item containing a one- to seven-character terminal name, left-justified, with binary-zero or blank fill. The terminal name may also be specified with a computational-1 item with binary zero fill. If the item contains only binary zero, the message is sent to the originating terminal. This terminal name must be a name appearing in the NETWid file (network file) which is validated for the same data base as the task issuing the SEND.† A terminal name consisting of the character P with zero fill cannot be specified.
cease-flag	Computational-1 item whose value is nonzero if the task is to end after the message is sent. If the value is zero, the task continues after this operation. The default value is 0.
output-flag	Computational-1 item whose value is nonzero if another SEND is to follow this one before a WAITINP or CEASE. If the value is zero, additional SENDS are not issued. The default value is 0. This parameter has meaning only for the stimulator.
block	Not required unless length parameter is used. If used, block must be a computational-1 item whose value is zero.
status	Not required unless length parameter is used. If used, stat must be a computational-1 item whose value is zero.
length	Length in characters of message. Use this parameter only when the data name of parameter mesaddr does not correctly reflect the desired message length. This parameter length must be a computational-1 item. Length is an optional parameter. When length is used all previous parameters must be specified.

NOTE

Under COBOL, SEND must be enclosed in quotation marks, because SEND is a reserved word for that compiler.

FORTRAN

CALL SEND(mesaddr,length,terminal,cease-flag, output-flag)

mesaddr	Name of the array containing the message to be sent.
length	Integer which specifies the length of the message characters.
terminal	Parameter whose value is a one- to seven-character terminal name left-justified with binary zero or blank fill to a word boundary. It specifies the terminal to which the message is sent. This terminal name must be a name appearing in the NETWid file which is validated for the same data base as the task issuing the send.† If terminal is of integer type with value zero or is not specified, the message is sent to the originating terminal.
cease-flag	Nonzero integer if the task is to end after the message is sent. If the integer value is zero, the task continues after this operation. The default value is 0.

†Tasks associated with application SY can SEND to any valid TAF terminal.

output-flag — Nonzero integer if another SEND is to follow this one before a WAITINP or CEASE. If the integer value is zero, additional SENDs are not issued. This parameter has meaning only for the stimulator.

COMPASS

SEND addr

addr — Address of a parameter table in the format shown in figure 2-10.

RELSCB Request

The RELSCB request releases all overflow communication blocks† to the system. After an RELSCB request is processed, subsequent LOADCB requests return a status word of 0.

The format is:

COBOL

ENTER RELSCB.

There are no parameters.

FORTRAN

CALL RELSCB

There are no parameters.

COMPASS

RELSCB

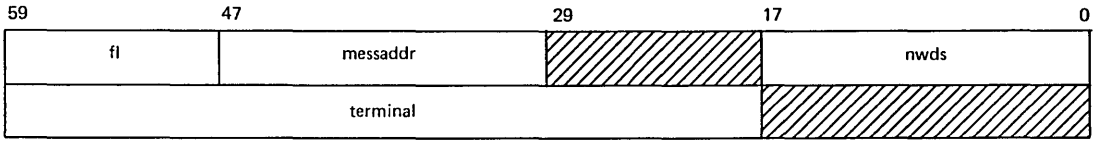
There are no parameters.

TSIM Request

The TSIM request provides limited interrogation of the terminal status table. This table, in central memory, is derived from the network file (refer to section 8). The TSIM request can search four fields in the table.

- Data base name field
- User argument field
- Communication line field
- Terminal name field

A list of names of those terminals which meet a defined criterion is returned to the user. The field specified by code is examined in each terminal table entry by taking the logical product of the field and mask to isolate a portion of the field. The logical difference of this product and the crit value is then obtained. If this result is zero, the terminal entry is placed in list. No action occurs if it is not zero; that is, the user does not meet the specified criterion. For example, an application program could use this to obtain a list of the terminals which are allowed to use a particular data base by selecting the data base field to be searched for a certain name.



fl — If bit 59 is set, message is sent to specified terminal; if not set, message is sent to originating terminal.

If bit 58 is set, task is terminated after the message is sent; if not set, task continues after this operation.

If bit 57 is set, additional SENDs are issued before a CEASE or WAITINP; if not set, additional SENDs are not issued. This is required only with the stimulator (refer to the System Maintenance Reference Manual).

mesaddr — Address of the message to be sent. The message must be terminated by 12 to 66 bits of binary zero as an end of line byte if using display code.

nwds — Length of the message in words, including the end of line byte.

terminal — Terminal name, 1 to 7 characters with binary zero fill if bit 59 is set. If bit 59 is not set, message is sent to the originating terminal.

Figure 2-10. SEND Parameter Table

†Described under LOADCB request.

The format is:

COBOL

ENTER TSIM USING code mask crit rleng list leng.

code — Computational-1 item whose value indicates which field is to be searched:

value	field
0	Data base name field
1	User argument field
2	Communication line field
3	Terminal name field

mask — Parameter whose value is taken as a binary mask. It is used in compiling the list of terminal names.

crit — Criterion value for the search.

rleng — Computational-1 item whose value TAF sets to indicate the number of entries found.

list — Item in which TAF lists found entries. If zero or absent, no list is returned but rleng is given as the number of found entries.

leng — Computational-1 item whose value indicates the number of words that list can hold. If zero or absent, no list is returned but rleng is given as the number of found entries. Only the lowest order 12 bits of this value are used.

FORTRAN

CALL TSIM(code,mask,crit,rleng,list,leng)

code — Integer whose value indicates which field is to be searched:

value	field
0	Data base name field
1	User argument field
2	Communication line field
3	Terminal name field

mask — Parameter whose value is taken as a binary mask. It is used in compiling the list of terminal names.

crit — Criterion value for the search.

rleng — Integer variable whose value TAF sets to indicate the number of entries found. This value is also returned as a FORTRAN function value.

list — Name of an array in which TAF places a list of found entries. If zero or absent, no list is returned, but rleng is given as the number of found entries.

leng — Integer variable whose value indicates the number of words that list can hold. If zero or absent, no list is returned, but rleng is given as the number of found entries. Only the lowest order 12 bits of this value are used.

COMPASS

TSIM addr

addr — Address of a parameter table in the format shown in figure 2-11.

TARO Request

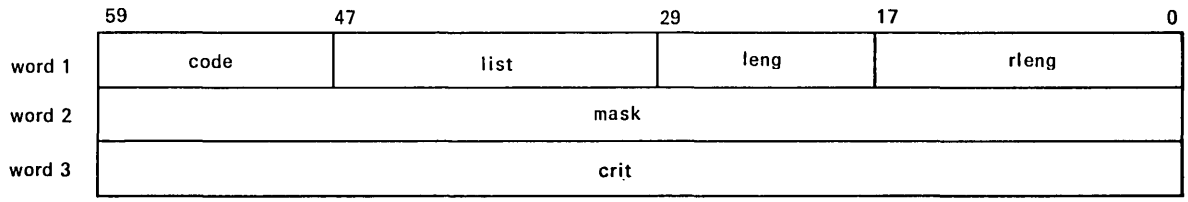
The TARO request allows the task to manipulate the user argument field in the terminal status table entry of a particular terminal. (Refer to section 8 for further information.) The user argument field (24 bits) is operated on in the following manner. The logical product of the user argument field and the lower 24 bits of mask are taken; then the logical difference of this result and value is determined. This value is placed in return and replaces the old user argument field. This allows the user to test a specific field to obtain its value and to establish a particular value in the field. For example, bit 1 of the user argument field may be used to establish a terminal recovery condition. The TARO request operates in the following manner.

Case 1: The request interrogates the user argument field and determines whether the recovery bit (bit 1) is set. The value of this bit in the user area is not changed and is placed in return. (Only bits 0 through 3 are shown in this example.)

logical	010	user argument field of TST
product	010	mask parameter
logical difference	010 000	value parameter
	010	return value = new user argument field

Case 2: The request sets the recovery bit.

logical	000	user argument field
product	010	mask parameter
logical difference	000 010	value parameter
	010	return value = new user argument field



code — Integer whose value indicates which field is to be searched:

value	field
0	Data base name field
1	User argument field
2	Communication line field
3	Terminal name field

list — Location of list to contain found entries. If zero or absent, no list is returned, but rleng is given as the number of found entries.

leng — Number of words that list can hold. If zero or absent, no list is returned, but rleng is given as the number of found entries. Only the lowest order 12 bits of this value are used.

rleng — The address at which TAF returns the number of entries found.

mask — Value to be taken as a binary mask. It is used in compiling the list of terminal names.

crit — Criterion value for the search.

Figure 2-11. TSIM Parameter Table

Case 3: The request clears the recovery bit.

logical product	010 000	user argument field mask parameter
logical difference	000 000	value parameter
	000	return value = new user argument field

The format is:

COBOL

ENTER TARO USING value mask terminal return.

value — Item whose value is a 24-bit value to be used to alter the user argument field.

mask — Parameter whose value is taken as a binary mask. Only the lowest 24 bits are used.

terminal — A 01-level item containing a one- to seven-character alphanumeric terminal name. The terminal name can also be a computational-1 item containing a binary zero (designating the originating terminal).

return — Item whose value TAF sets to indicate the resultant user argument bits.

FORTRAN

CALL TARO(value,mask,terminal,return)

value — Parameter whose 24-bit value is used to alter the user argument field.

mask — Parameter whose value is taken as a binary mask. Only the lowest 24 bits are used.

terminal — Parameter whose value is a one- to seven-character alphanumeric terminal name, containing either blank or binary zero fill to a word boundary. If this parameter is omitted or is a binary zero, then the originating terminal is used.

return — Parameter in which TAF places the resultant user argument bits. This value is also optionally returned as the value of a FORTRAN function.

COMPASS

TARO addr

addr — Address of a parameter table in the format shown in figure 2-12.

IIO Request

The IIO request allows a task to change its own I/O limit as measured in RA+1 calls from the task. Any request given in this manual or any data manager request is considered an RA+1 call. The default value is 320 calls.

The format of the request is:

COBOL

ENTER IIO USING nio.

nio — Computational-1 item whose value specifies the new I/O limit.

FORTRAN

CALL IIO(nio)

nio — Integer whose value specifies the new I/O limit.

COMPASS

IIO addr

addr — Address of a parameter table in the format shown in figure 2-13.

BWAITINP Request (COMPASS Only)

The BWAITINP request allows a task to move data from a terminal to a task field location. This location has the same format as the communication block (refer to figure 2-1). The request is processed the same as a WAITINP request, except that when the terminal data is moved to the specified location, it is moved to the area specified by the supplied parameter (addr), not the first word address of the task (as in the WAITINP request). The first word address is not changed by the execution of this request.

The format is:

BWAITINP addr

addr — Address of first word of the storage area where the terminal data is to be moved.

TAF does not pass this terminal data to tasks called with the CALLTSK and CALLRTN requests.

TPSTAT Request (COMPASS Only)

The TPSTAT request allows a task to determine whether it is running under TS or NAM. Terminal control is supported differently, depending on the terminal subsystem. The user can subsequently change the output data to its correct form for the terminal subsystem.

The format is:

TPSTAT addr

addr — Address where the identifier of the active terminal subsystem is to be returned as shown in figure 2-14.

BEGIN Request

The BEGIN request allows a task to specify the area where the terminal input data is to be placed, if an area other than the first word address of the task field length is desired. It supports programming languages which do not reserve the first word address of the task for the common block that is used for the communication block. The task must be installed in the appropriate task library with the (solicited) input directive, because the default is unsolicited. If task chains are involved, the calling task must issue a BEGIN request before resuming execution of the original task, because the calling task must be able to obtain the communication block containing the returned data from the called task. The returned communication block in solicited mode is in the same form as in unsolicited mode. Unless there is a specific need for this request, its use is not recommended because more execution overhead is required to support the secondary load request.

The format is:

COBOL

ENTER BEGIN USING buff.

buff — A 01-level item identifying the user-defined buffer area.

FORTRAN

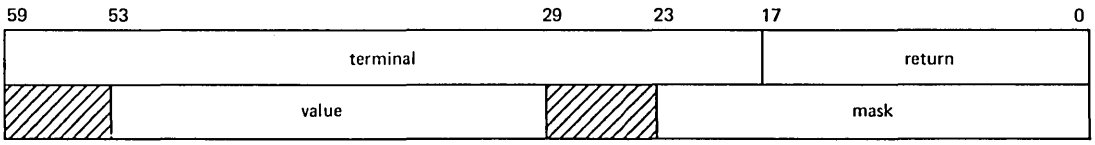
CALL BEGIN(buff)

buff — An array name identifying the user-defined buffer area.

COMPASS

BEGIN addr

addr — Address of the user-defined buffer area.



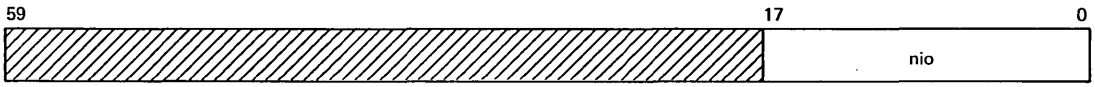
terminal — One- to seven-character terminal name, left-justified with binary zero fill, for which the operation is performed. If terminal is a binary zero, then the originating terminal is used.

return — Location in which to place the resultant user argument bits.

value — 24-bit value to be used to alter the user argument field.

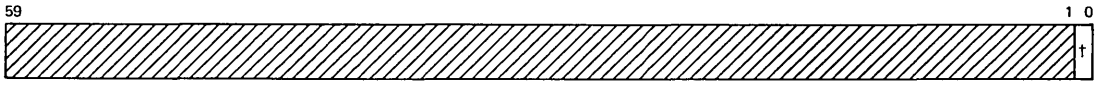
mask — Value to be taken as a binary mask. Only the lowest 24 bits are used.

Figure 2-12. TARO Parameter Table



nio — New I/O limit.

Figure 2-13. IIO Parameter Table



t	Meaning
0	TS active (TPTLX)
1	NAM active (TPNAM)

The symbols TPTLX and TPNAM are defined in the assembly text COMKMAC.

Figure 2-14. TPSTAT Return Values

TASK CONTROL REQUESTS

These requests are used for task control.

- RECALAL

  Pause until all data manager requests are complete.

- CEASE

  End a task.

- ITL

  Set a task time limit.

- LOGT

  Log a terminal out of TAF.

- KPOINT

  Initiate execution of K display commands.

- WAIT

  Suspend processing for a given amount of time.

RECALAL Request

The RECALAL request causes a pause in task execution until all data manager activity previously initiated for the task is complete. This request is for use with the TAF data manager.

The format is:

COBOL

ENTER RECALAL.

There are no parameters.

FORTRAN

CALL RECALAL

There are no parameters.

COMPASS

RECALAL

There are no parameters.

CEASE Request

The CEASE request provides task end processing and transaction completion processing. If a predetermined task list exists for the chain (that is, subsequent tasks have been requested for scheduling, or the CEASE is a return from a CALLRTN request), no special processing occurs on a CEASE request from a task. However, if the request is from the final task in the chain, the transaction executive releases all communication blocks and notifies the data manager that updated records should be replaced on the data base. The one exception to this procedure occurs if a task selects an abnormal CEASE by use of an optional parameter.

A task which terminates in this manner is automatically the final task in the chain. After an abnormal CEASE occurs, all communication blocks are released, and records updated by the task but not yet updated on the data base are not replaced on the data base.

The format is:

COBOL

ENTER CEASE USING a.

a Computational-1 item whose value indicates the type of CEASE selected:

a	Type of CEASE
0	Normal CEASE.
1	Abnormal CEASE without memory dump.
Any other	Abnormal CEASE with memory dump.

FORTRAN

CALL CEASE(a)

a Integer whose value indicates the type of CEASE selected:

a	Type of CEASE
0	Normal CEASE.
1	Abnormal CEASE without memory dump.
Any other	Abnormal CEASE with memory dump.

COMPASS

CEASE a

a Integer whose value indicates the type of CEASE selected:

a	Type of CEASE
0	Normal CEASE.
1	Abnormal CEASE without memory dump.
Any other	Abnormal CEASE with memory dump.†

ITL Request

The ITL request allows a task to set the time limit for the task to a new value. The time limit is expressed in exchange jump time units, that is, the number of times a task executed its full time slice without interruption. In addition, each call to this function decreases the CPU priority of the task by 1 until the lower limit of priority is reached to ensure good service to other tasks. The time limit default value is a product of two installation parameters. The released value is 1.12 CPU seconds. If this is exceeded, the task aborts, and the message TIME LIMIT EXCEEDED is issued to the terminal.

†Dumps are identified by the transaction subsystem user number or DSDUMP parameters (refer to section 5).

The format is:

COBOL

ENTER ITL USING ntm.

ntm Computational-1 item whose value specifies the new time limit.

FORTRAN

CALL ITL (ntm)

ntm Integer whose value specifies the new time limit.

COMPASS

ITL addr

addr Address of a parameter table in the format shown in figure 2-15.

LOGT Request (COMPASS Only)

The LOGT request logs out a terminal from TAF and returns it to nontransaction mode. The terminal can then access other timesharing applications. A sample task is provided using this request. (Refer to System Tasks in section 10.)

The format is:

COMPASS

LOGT

There are no parameters.

KPOINT Request

The KPOINT request allows a task to initiate execution of any K display command. Section 6 describes three K display commands: K.DUMP, K.DUMPLIM, and K.DSDUMP.

K display commands are normally issued from the system console to communicate with and to request specific actions of TAF. K display commands should be issued with care and then only when the results of the command are well-defined.

When a task issues a K display command, the prefix K. is omitted. For example, the entry from the console is K.DUMP., but the same command issued from a task is DUMP..

All K display commands, except K.DUMP, require that the calling task be on the system task library. Any task can issue the K.DUMP command.

The format is:

COBOL

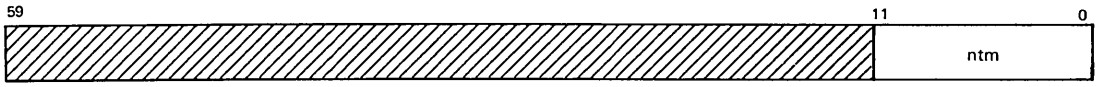
ENTER KPOINT USING addr

addr 01-level name of the character string (K display command). The length of the character string cannot exceed 80 characters. The string must begin on a word boundary and must be terminated by a period or a right parenthesis; the exception is the K.MESSAGE command, which must be terminated by 6 bits of zeros after either a period or a right parenthesis.

FORTRAN

CALL KPOINT(addr)

addr Address of the character string (K display command). The length of the character string cannot exceed 80 characters. The string must begin on a word boundary and must be terminated by a period or a right parenthesis; the exception is the K.MESSAGE command, which must be terminated by 6 bits of zeros after either a period or a right parenthesis.



ntm New time limit.

Figure 2-15. ITL Parameter Table

COMPASS

KPOINT addr

addr Address of the character string (K display command). The length of the character string cannot exceed 80 characters. The string must begin on a word boundary and must be terminated by a period or a right parenthesis; the exception is the K.MESSAGE command, which must be terminated by 6 bits of zeros after either a period or a right parenthesis.

WAIT Request

The WAIT request provides tasks with the ability to have TAF suspend processing for a specified number of seconds (0 to 3600). The transaction must have no outstanding data manager requests at the time a WAIT request is issued.

The format is:

COBOL

ENTER WAIT USING time.

time A Computational-2 data item which indicates the number of seconds for which processing is to be suspended. If unspecified or out of range, the task is aborted. Only the integer portion is significant.

FORTRAN

CALL WAIT(time)

time A real number or variable which indicates the number of seconds for which processing is to be suspended. If unspecified or out of range, the task is aborted. Only the integer portion is significant.

COMPASS

WAIT addr

addr The address of a word specifying the number of seconds (N) for which processing is to be suspended. The word has the form: 48/0,12/N. If unspecified or out of range, the task is aborted.

TASK UTILITY REQUESTS

The following task utility requests provide capabilities for the FORTRAN and COBOL user, some of which are difficult to do because they are at the bit level.

- LOGIN

  Determine if a specified terminal is logged in.

- EXTRACT

  Extract a bit string from a word.

- INSERT

  Insert a bit string into a word.

LOGIN Request

The LOGIN request allows a task to determine if a specified terminal is logged in before it issues a SEND request to that terminal. TAF aborts a task that issues a SEND request without recall to a terminal that is not logged in.

NOTE

It is possible for the receiving terminal to log out after the task issues a LOGIN request and before the task issues a SEND request. Therefore, it is recommended that the task issue the SEND request soon after issuing the LOGIN request.

The format is:

COBOL

ENTER LOGIN USING terminal status.

terminal A one- to seven-character alphanumeric terminal name, left-justified. A binary zero specifies the terminal associated with the transaction that is running.

status Address of the location to contain the returned status. The format of the returned status is computational-1. Values for status are:

0 Terminal not logged in.

1 Terminal logged in.

FORTRAN

CALL LOGIN (terminal, status)

terminal A one- to seven-character alphanumeric terminal name, left-justified. Possible forms are nHf and nLf (refer to the FORTRAN Extended Reference Manual). A binary zero specifies the terminal associated with the transaction that is running.

status Address of the location to contain the returned status. The format of the returned status is integer. Values for status are:

0 Terminal not logged in.

1 Terminal logged in.

EXTRACT Request

The EXTRACT request allows a task to extract a bit string from a word and move it to another word, right-justified. For instance, the EXTRACT request can be used to check fields in the terminal status table or the communication block.

In the following request formats, if n is not within the specified range, if bbp plus 1 minus n is less than zero, or if there are any other errors in the parameters, TAF aborts the task and issues the following message.

TASK REQUEST ARGUMENT ERROR.

The format is:

COBOL

ENTER EXTRACT USING loc1 loc2 bbp n.

- loc1 — Data name of the source field. Any data type is allowed. loc1 must begin on a word boundary.
- loc2 — Data name of the computational-1 destination field. The bit string is right-justified in this field.
- bbp — Beginning bit position of field loc1; this is the leftmost bit of the field. Bits are numbered 59 through 0.
- n — Number of bits to extract; the range of values for n is 1 to 59, inclusive. The format of n must be computational-1.

FORTRAN

CALL EXTRACT (loc1,loc2,bbp,n)

- loc1 — Name of the integer source field.
- loc2 — Name of the integer destination field. The bit string is right-justified in this field.
- bbp — Beginning bit position of field loc1; this is the leftmost bit of the field. Bits are numbered 59 through 0.
- n — Number of bits to extract; the range of values for n is 1 to 59, inclusive. n must be integer type.

The COBOL STRING and UNSTRING statements also provide capabilities for manipulating character-aligned data. The EXTRACT request should be used only when the use of these COBOL statements is impossible or inconvenient.

INSERT Request

The INSERT request allows a task to insert a bit string into a word. For instance, the INSERT request can be used to set fields in the terminal status table.

In the following formats, if n is not within the specified range, if bbp plus 1 minus n is less than zero, or if there are any other errors in the parameters, TAF aborts the task and issues the following message.

TASK REQUEST ARGUMENT ERROR.

The format is:

COBOL

ENTER INSERT USING loc1 loc2 bbp n.

- loc1 — Data name of the computational-1 source field. The bit string must be right-justified in this field.
- loc2 — Data name of the destination field. Any data type is allowed. loc2 must begin on a word boundary.
- bbp — Beginning bit position of field loc2; this is the leftmost bit of the field. Bits are numbered 59 through 0.
- n — Number of bits to insert; the range of values for n is 1 to 59, inclusive. The format of n must be computational-1.

FORTRAN

CALL INSERT (loc1,loc2,bbp,n)

- loc1 — Name of the integer source field. The bit string must be right-justified in this field.
- loc2 — Name of the integer destination field.
- bbp — Beginning bit position of field loc2; this is the leftmost bit of the field. Bits are numbered 59 through 0.
- n — Number of bits to insert; the range of values for n is 1 to 59, inclusive. n must be integer type.

The COBOL STRING and UNSTRING statements also provide capabilities for manipulating character-aligned data. The INSERT request should be used only when the use of these COBOL statements is impossible or inconvenient.

CDCS can be used alone or with one other data manager (CRM, TOTAL,or TAF) in the same transaction. If a task is to use CDCS, the CDCS Reference Manual should be consulted for information on how to set up the data base (schema, subschema, master directory, and files). This data base is associated with CDCS and the files will be at the control point of CDCS during execution. Tasks can use the CDCS verbs for data base access, provided that they are compiled with the appropriate subschema relating them to a CDCS data base.

In the following, xx represents the two-character data base identifier.

A TAF application (known by the xx identifier in the TCF) is linked to a CDCS data base by means of the subschemas used in the tasks associated with the application (these are the tasks on xxTASKL). All tasks in a transaction must use the same subschema or a fatal task error (detected by CDCS) will occur.

A transaction will be considered to be connected to CDCS if it has successfully issued a request to CDCS and the terminate request to CDCS has not been completed. (A CDCS terminate request from other than the last task in a chain will be ignored.) If the last task in the chain does not issue a CDCS terminate request, one will be issued automatically when the last task completes.

A task will abort if it attempts to use CDCS when CDCS is not present in the system or if CDCS aborts while a transaction is connected to it. If a transaction aborts while it is connected to CDCS, TAF will issue an abnormal terminate notice to CDCS for the transaction (unless the abort originated with CDCS).

The following TAF files are associated with an application using CDCS:

- TCF, xxJ (required)
- xxTASKL (required unless all tasks are on system task library)
- xxJOR1, xxJOR2, xxJOR3 (optional)

The possibility of a deadlock involving CDCS and another data manager (TAF, TOTAL,or CRM) exists. For instance, if task X locks some resources under one data manager and then tries to lock other resources under another data manager which task Y initially locked, then the two tasks may be deadlocked if task Y tries to lock any of the resources locked by task X. It is the responsibility of the installation, application writers, and data base administrator to prevent these situations from occurring. Such deadlocks can be prevented by locking resources in a specified order only, for example, lock in data manager A resources first, then data manager B resources, and so forth.

RESTRICT clauses cause tasks to automatically request additional central memory. All of the advice and requirements relevant to a task requesting that its FL be increased apply when RESTRICT clause processing is used (refer to MEMORY Request, section 2).

This section is intended for the data base administrator and can be omitted by the applications programmer.

This section discusses three topics: the xxJ file, journal files, and the TAF configuration file (TCF).

An xxJ file is associated with each data base and has the following functions.

- Defines the user's user number, password, and user index.
- Defines optional journal files.
- Defines optional device residence for the task library and some files specific to the individual data managers (refer to the xxJ file descriptions which follow).

Up to three journal files can be specified optionally to supplement the system journalizing. A task can write to one of the journal files by using the JOURNL request.

TCF, the TAF configuation file, defines the data bases associated with a data manager.

A procedure file (TAFxxxx) is used to initialize TAF. Control statements may be added to it for recovery or to establish the operating environment. For example, control statements which attach trace files (xxTFIL),† pool files (xxERPF),† journal files (xxJORn), and data base files can be included in this file. Refer to the NOS Installation Handbook for more information (refer to preface for publication number).

xxJ FILE

An indirect access permanent file, xxJ, where xx is the data base name, is required for each data base. If additional journal files are needed, they are defined in the xxJ file. The xxJ file is saved under the transaction subsystem user number. The file is in card image format and varies depending on the data manager used.

NOTE

The transaction subsystem user number should be under the control of a data base administrator. The applications programmer should not have access to this number.

TAF DATA MANAGER xxJ FILE

The following shows the structure of the xxJ file when using the TAF data manager.

```
xxJ

USER(usernum,password,familyname)††

UI=n.

EDT,PN=packname,R=device.
xxJOR1,dv,ft.
xxJOR2,dv,ft.                    } Optional.
xxJOR3,dv,ft.
xxTASKL,PN=packname,R=device.
```

The parameters for the xxJ file are:

xx

Data base name.

n

User index (obtained from site analyst). This statement is optional.

packname

Packname of the auxiliary device on which the user's element descriptor table (EDT) or xxTASKL (user's task library) will reside.

device

Type of device on which the user's file will reside; for example, DI (for 844), or DI3 (for a multiunit 844 consisting of three devices).

dv

Device type:

MT Magnetic tape

MS Mass storage

ft

File type (figure 4-1):

B Buffered write

R Nonbuffered write

†Refer to the TAF/TS Data Manager Version 1 Reference Manual for information on these files.

††Unless an EDT statement is used, TAF locates the data base using the family name parameter in the procedure file that initializes TAF and the usernum and password parameters in the xxJ file. The usernum, password, and family name parameters are used for validation.

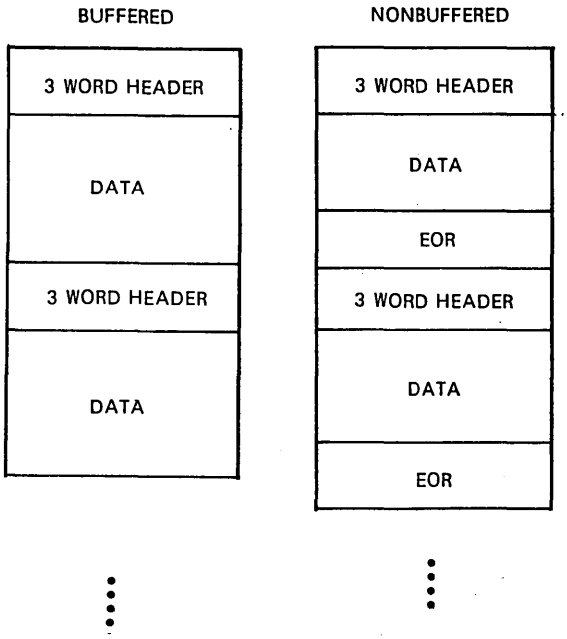


Figure 4-1. Journal File
(Buffered and Nonbuffered Structure)

CRM DATA MANAGER xxJ FILE

The following shows the structure of the xxJ file when using the CRM data manager.

```
xxJ

USER(usernum,password,
  familyname)†

xxJOR1,dt,ft.
xxJOR2,dt,ft.
xxJOR3,dt,ft.
xxTASKL,PN=packname,
  R=device.                    } Optional.

CRM(xxpfnl,type,mode,
  users,locks,mrl,kl,
  hash,packname,device)
        .
        .
        .
CRM(xxpfnn,type,mode,
  users,locks,mrl,kl,
  hash,packname,device)        } One CRM statement for each file.
```

The parameters are:

xx — Data base name.

dt — Device type:

- MS Mass storage
- MT Seven-track tape
- NT Nine-track tape

ft — File type: (figure 4-1)

- B Buffered write
- R Nonbuffered write

packname — Pack name of the auxiliary device on which the user's task library or data base files reside. This parameter is optional on the CRM statement.

device — Type of device on which the user's task library or data base files reside. For example, DI (for an 844) or DI3 (for a multiunit 844 consisting of three devices). This parameter is optional on the CRM statement.

xxpfni — Two- to seven-character file name. xx is the data base name. pfni is the zero- to five-character data base file name.

type — CRM file type:

- DA Direct access
- IS Indexed sequential

mode — Attach mode of file xxpfni:

- M Modify, append, or read
- R Read
- RM Read, with another user concurrently attaching the file in mode M
- W Write, modify, append, or read

The mode parameter controls access to the file to ensure that either TAF or a batch job can write on or modify the file at a given time. However, TAF and a batch job can read the file simultaneously. Refer to the ATTACH statement in the NOS Reference Manual, volume 1 for access modes granted for different combinations of multiple access.

NOTE

If the TAF procedure file attaches the file, TAF uses that mode.

users — Number of transactions allowed to open file xxpfni concurrently. When TAF terminates, the TAF dayfile contains a message that gives the number of OPEN requests not granted because the value of the users parameter was exceeded. Each user uses 38 words plus the kl parameter size (in words).

†TAF locates the data base using the familyname parameter in the procedure file that initializes TAF and the usernum and password parameters in the xxJ file. The usernum, password, and familyname parameters are used for validation.

locks — Number of records the transaction can lock for file xxpfni. When TAF terminates, the TAF dayfile contains a message that gives the number of locks not granted because the value of the locks parameter was exceeded. Each lock uses two words plus the kl parameter size (in words).

mrl — Number specifying the maximum record length (in characters) for file xxpfni.

kl — Number specifying the length (in characters) of the key for file xxpfni.†

hash — One- to seven-character indirect access file that contains the binary code of the hashing routine for file xxpfni. The entry point for the hashing routine must be this file name (hash). If this parameter is omitted, CRM uses the hashing routine supplied with CRM. This parameter applies only for CRM direct access files.

NOTE

This indirect access file must be stored under the usernum parameter on the USER statement.

TOTAL DATA MANAGER xxJ FILE

The following shows the structure of the xxJ file when using the Total data manager.

```
xxJ
USER(usernum,password,familyname)††
UI=n.                                  }
xxJOR1,dv,ft.                          }
xxJOR2,dv,ft.                          }  Optional.
xxJOR3,dv,ft.                          }
xxTASKL,PN=packname,R=device.          }
NL.                                    }
LG,PN=packname,R=device.               }  Either the NL statement or the LG statement must be selected. PN and R parameters are optional. NL causes Total not to log recovery data. LG causes Total to log recovery data.
aaaa,PN=packname,R=device.             }
        .                              }
        .                              }  PN and R parameters are optional.
        .                              }
zzzz,PN=packname,R=device.             }
```

The parameters for the xxJ file are:

xx — Data base name.

n — User index (obtained from site analyst).

dv — Device type:

- MS Mass storage
- MT Seven-track tape
- NT Nine-track tape

ft — File type (refer to figure 4-1):

- B Buffered write
- R Nonbuffered write

packname — Packname of the auxiliary device on which the user's task library (xxTASKL) or data sets will reside.

device — Type of device on which the user's file will reside for example, DI (for 844) or DI3 (for a multiunit 844 consisting of three devices).

aaaa
.
.
.
zzzz — Four-character data set name.

CDCS OR NO DATA MANAGER xxJ FILE

If an application uses CDCS and one of the other data managers (TAF, CRM, TOTAL), then the xxJ file should be structured as specified for the other data manager.

If an application accesses only the CDCS data manager or no data manager, the structure of the xxJ file is as follows:

```
xxJ
USER(usernum,password,familyname)††
xxJOR1,dt,ft.                          }
xxJOR2,dt,ft.                          }  Optional.
xxJOR3,dt,ft.                          }
xxTASKL,PN=packname,R=device           }
```

The parameters are:

xx — Application (data base) name.

dt — Device type:

- MS Mass storage
- MT Seven-track tape
- NT Nine-track tape

ft — File type (figure 4-1)

- B Buffered write
- R Nonbuffered write

packname — Pack name of the auxiliary device on which the user's task library resides.

device — Type of device on which the user's task library resides.

†The user can specify octal numbers and decimal numbers by using a postradix of B and D, respectively. Decimal is the default.

††TAF locates the data base using the familyname parameter in the procedure file that initializes TAF and the usernum and password parameters in the xxJ file. The usernum, password, and familyname parameters are used for validation.

CREATING xxJ FILE

To enter an xxJ file, use the following procedure at a terminal:

1. Log in under the transaction subsystem user number.
2. Enter NEW,xxJ ⓒⓡ (where xx is the data base name).
3. Enter TEXT ⓒⓡ ; the system will respond ENTER TEXT MODE.
4. Enter the data given previously; the system gives no response.
5. Press BREAK key; the system will respond EXIT TEXT MODE PACK COMPLETE.
6. Enter SAVE,xxJ ⓒⓡ; the file will be made permanent.

JOURNAL FILES

The two types of journal files are system and data base.

SYSTEM JOURNAL FILE

The system journal file (a direct access file named JOUR0 which may be defined under the transaction subsystem user number or may be set up by the TAF procedure file at initialization) is not associated with any particular data base. All transaction data (input from terminals), errors, and transaction completion notification is automatically recorded on JOUR0. The terminal status table is also journalized periodically. Data sent to terminals by tasks is not recorded on JOUR0. JOUR0 will be created at initialization if not already present under the transaction facility username.

DATA BASE JOURNAL FILES

The transaction subsystem allows up to three user-specified direct access journal files for each data base. The allowable names are xxJOR1, xxJOR2, and xxJOR3, where xx is the data base name. At initialization, TAF checks the xxJ files associated with all active applications to determine which xxJORn files are needed. It will then attach any specified xxJORn files not already at the control point and attempt to define any which it cannot find. The transaction subsystem user number must have write permission on these files. When a task makes a journal request (refer to JOURNL request), the data base for which the originating terminal is validated is used to determine the journal file on which to write. If the task is associated with a terminal validated for all data bases, the data base specified on the last data base request is used to determine the proper journal file.

JOURNAL FILE ENTRY HEADER

Each journal file entry (refer to JOURNL request) has a header in the format shown in figure 4-2.

JOURNL REQUEST

The JOURNL request allows file entries to be made to supplement automatic journalizing.

The format is:

COBOL

ENTER JOURNL USING jnum message.

jnum — Computational-1 item whose value indicates the number of the journal file to which the entries are to be made. The value of this parameter must be right-justified with zero fill.

jnum	Journal File
0	JOUR0 (the system journal file to which the transaction executive normally makes journal entries).
1	Data base journal file xxJOR1.
2	Data base journal file xxJOR2.
3	Data base journal file xxJOR3.

message — Name of the 01-level data item containing the message to be journalized.

FORTRAN

CALL JOURNL(jnum,message,length)

jnum — Integer whose value indicates the number of the journal file to which the entries are to be made.

jnum	Journal File
0	JOUR0 (the system journal file to which the transaction executive normally makes journal entries).
1	Data base journal file xxJOR1.
2	Data base journal file xxJOR2.
3	Data base journal file xxJOR3.

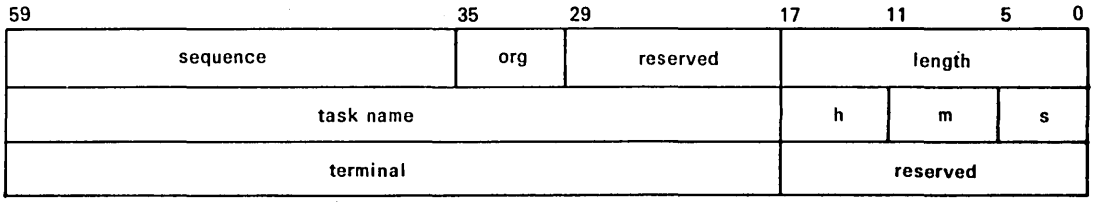
message — Name of the array containing the message to be journalized.

length — Integer whose value specifies the length of the message in characters.

COMPASS

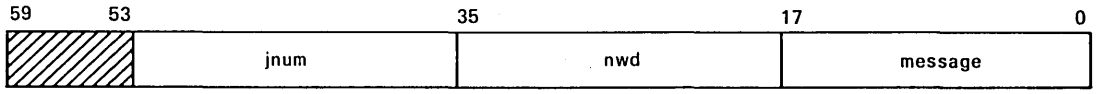
JOURNL addr

addr — Address of a parameter table in the form shown in figure 4-3.



sequence	Transaction sequence indicator
org	Origin indicator (octal):
	0 Task origin (JOURNL request)
	1 Transaction subsystem origin (input)
	2 Data manager origin
	3 Transaction subsystem recovery/statistical data
	4 End-of-transaction indicator
	5 Incomplete block of terminal input data
	6 Terminal input for an interactive task
	7 Illegal intercontrol point transfer
	10 On-line LIBTASK update (TT option)
	11 CDCS detected error (words following header contain error message)
length	Length of entry (equals header plus data block)
task name	Name of task if task origin requested
h m s	Packed time (hour, minute, second)
terminal	Name of terminal associated with transaction if task origin

Figure 4-2. Journal File Entry Header



jnum	Journal file number, right-justified with binary zero fill:
	0 JOUR0 (the system journal file to which the transaction executive normally makes journal entries)
	1 to 3 Data base journal file (xxJOR1, xxJOR2, or xxJOR3).
nwd	Length of message in words.
message	Address of message to be journalized.

Figure 4-3. JOURNL Parameter Format

If a journal file is defined on magnetic tape and an end-of-reel is detected, the transaction executive unloads the tape and checks for another tape on which to write. If the operator has not preassigned a unit, the journal file is placed on mass storage. If the operator enters the K.ASSIGN command, the transaction executive copies the mass storage journal file to the tape and places all subsequent entries on the tape. If the mass storage file does not fit on the tape, the message *UNABLE TO USE TAPE* is issued.

TAF CONFIGURATION FILE (TCF)

The TAF Configuration File (TCF) is a required indirect access permanent file saved under the transaction facility user name. It is used to specify which data managers are to be loaded by TAF and which applications (data bases) are to be associated with each data manager. Data in the TCF should be in the format:

DMS,dm,sw,xx_1,xx_2,...,xx_n.

DMS,dm,sw,xx_1,xx_2,...,xx_n.

DMS,dm,sw,xx_1,xx_2,...,xx_n.

DMS,dm,sw,xx_1,xx_2,...,xx_n.

Optional

The parameters on the DMSTAT statements have the following meanings:

Parameter	Meaning
dm	Name of the data manager. Must be TAF, CRM, TOTAL, or OTHER. The same data manager can be specified on any number of DMS statements. The dm value OTHER is to be used for specifying an application which either is not associated with a data manager or is associated only with the CDCS data manager.
sw	Status. Must be ON or OFF.
ON	The specified data manager (TAF, TOTAL, or CRM) will be loaded. The data bases specified on the same statement are to be made available for TAF processing. These data bases are referred to as active.
OFF	Data bases specified on the same statement are not made available for TAF processing.
xx_i	Data base identifier. Must not be duplicated on the same or any other active DMS statement. For the CRM and TAF data managers it is the two letter data base name. For the TOTAL data manager, this parameter has the format: xxyyyy which is the data base name (the DBMOD† file name) specified in the DATA-BASE-NAME† statement of the the DBDL† statements. xx is the unique TAF data base name and yyyy is any combination of four alphanumeric characters.

A separate DMS statement for CDCS (OTHER) is not necessary if CDCS is to be used in conjunction with another data manager.

The TCF file must contain at least one DMS statement, and each DMS statement with status set to ON must have at least one data base identifier or TAF will abort.

The maximum number of data bases which may be associated with one data manager is specified by the installation parameter MAXDB. This value is 25 unless altered by your site.

The maximum length of each DMS statement is 160 display code characters.

†Refer to the Total-CDC Version 1 Reference Manual for a detailed description.

The transaction subsystem interfaces with the batch subsystem in two ways. First, batch jobs can initiate tasks by use of the BTRAN request. Second, tasks can route jobs to the batch input queue by use of the SUBMT request.

The BTRAN and SUBMT requests are described in this section.

BTRAN REQUEST

The BTRAN request enables a batch job to start a transaction. The transaction is passed to the transaction executive through the intercontrol point mechanism; therefore, the user must be validated for intercontrol point transfers (AW=CSTP). The user passes a transaction to the transaction executive via intercontrol point area 1 with the header word shown in figure 5-1.

The format is:

COBOL

ENTER BTRAN USING tran status.

tran — Name of the 01-level item containing the header word for the transaction to be submitted.

status — Name of the data that contains the returned status; the status is returned as a computational-1 value.

Octal status values are as follows:

1 Transaction submission is complete.

3 Transaction executive is not up.

7 Transaction data is too long; maximum length is 620 characters or less according to installation option.

11 This user cannot submit batch jobs to TAF. Bit 9 (CSTP) in the user's access word is not set. Refer to the MODVAL control statement in the System Maintenance Reference Manual for further information.

If the terminal name is invalid, the transaction executive sends the following message to the TAF dayfile.

ILLEGAL TERMINAL NAME

This message appears at the system console. Users receiving questionable results should contact the console operator to see if this message appeared while they were trying to submit a batch transaction.

The EXTRACT and INSERT requests (section 2), used in conjunction with computational-4 items, can be used to construct the data block necessary for this request (figure 5-1).

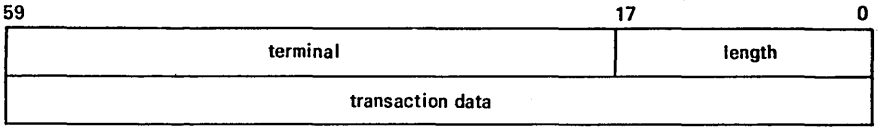
FORTRAN

The user calls the BTRAN request in two ways. The first method is a FORTRAN function.

i=BTRAN(tran)

i — Integer variable that contains the request status after BTRAN is called. The user must declare BTRAN as an integer.

tran — Name of the array containing the header word for the transaction to be submitted.



terminal — One- to seven-character terminal name, left-justified with zero fill, for which the operation is to be done.

length — Length of the transaction in words; includes the header word and the transaction data.

transaction data — Transaction name (or transaction name and input data) to be processed by TAF.

Figure 5-1. BTRAN Header Word

The second method is a subroutine call.

CALL BTRAN(tran,status)

tran Name of the array containing the header word for the transaction to be submitted.

status Address of the variable that contains the returned status; the status is returned as an octal integer value.

Octal status values are:

1 Transaction submission is completed.

3 Transaction executive is not up.

7 Transaction data is too long; maximum length is 620 characters or less according to installation option.

11 This user cannot submit batch jobs to TAF. Bit 9 (CSTP) in the user's access word is not set. Refer to the MODVAL control statement in the System Maintenance Reference Manual for further information.

If the terminal name is invalid, the transaction executive sends the following message to the TAF dayfile.

ILLEGAL TERMINAL NAME

This message appears at the system console. Users receiving questionable results should contact the console operator to see if this message appeared while they were trying to submit a batch transaction.

COMPASS

BTRAN tran

tran Address of the header word for the transaction to be submitted.

The status of the request is returned in register X6.

Octal status values are as follows:

1 Transaction submission is complete.

3 Transaction executive is not up.

5 The transaction executive is busy because the batch buffer is full, the subsystem is being moved, or the subsystem is job-advancing. Resubmit the request.

7 Transaction data is too long; maximum length is 620 characters or less according to installation option.

11 This user cannot submit batch jobs to TAF. Bit 9 (CSTP) in the user's access word is not set. Refer to the MODVAL control statement in the System Maintenance Reference Manual for further information.

If the terminal name is invalid, the transaction executive sends the following message to the TAF dayfile.

ILLEGAL TERMINAL NAME

Figure 5-2 demonstrates the use of BTRAN.

```
BTRANEX.
USER,XYZ1234,XYZ1.
CHARGE,1234,567Z890.
COBOL5.
LDSET(LIB=BDMLIB)
LOAD,LGO.
NOGO,LGOB.
LGOB.
- -EOR- -
       IDENTIFICATION DIVISION.
       PROGRAM-ID. BTRANEX.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. CYBER.
       OBJECT-COMPUTER. CYBER.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 BATCH-TRANSACTION.
           02 TERMINAL-NAME          VALUE IS "ABC1234"     PIC X(7).
           02 DATA-LENGTH            VALUE 2    COMP-4      PIC 9999.
           02 TRANSACTION-DATA       VALUE IS "EX.M12"      PIC X(6).
       01 BTRAN-STATUS               VALUE IS 0  COMP-1     PIC 9999.
       PROCEDURE DIVISION.
       START-RUN.
           ENTER BTRAN USING BATCH-TRANSACTION, BTRAN-STATUS.
           DISPLAY "STATUS=" BTRAN-STATUS.
           STOP RUN.
```

Figure 5-2. BTRAN Request

M12 is the name of the transaction to be processed by TAF. Output from SEND or DISPLAY statements in M12 will appear at terminal ABC1234, if it is logged in to TAF, unless otherwise directed within M12. STATUS= n goes to the OUTPUT file associated with the batch job.

SUBMT REQUEST

The SUBMT request allows a task to route a job to the local batch input queue. The address parameter in the call points to the area in the task field length that contains the job to be routed in control word format. This format is described in conjunction with the READCW macro in the NOS Reference Manual, volume 2 (refer to preface for publication number). Using this format, the job may consist of multiple records.

The format is:

COBOL

ENTER SUBMT USING addr.

addr Name of the 01-level item containing the job data to be routed. This data is in the format shown in figure 5-3.

FORTRAN

CALL SUBMT (addr)

addr Name of an array containing the job data to be routed. This data is in the format shown in figure 5-3.

COMPASS

SUBMT addr

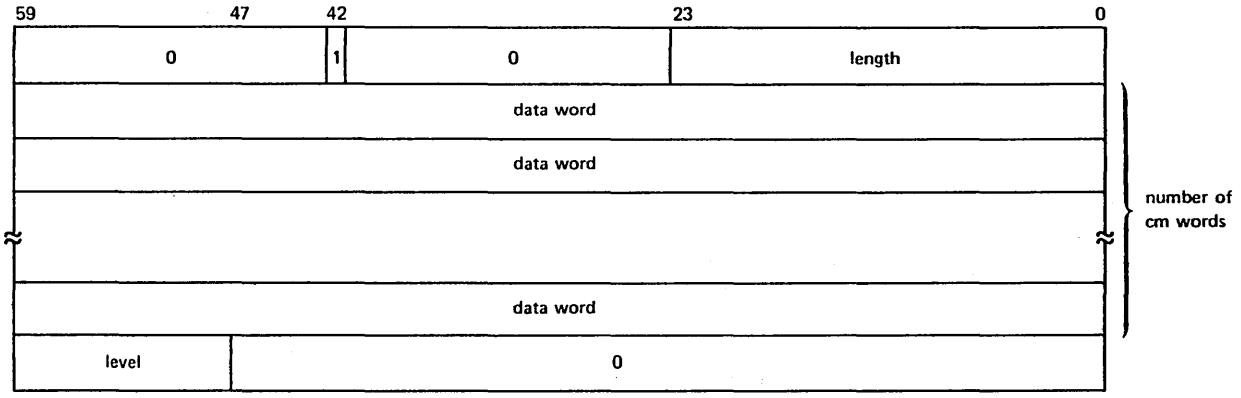
addr Address of the first word of the job data to be routed in control word format as shown in figure 5-3.

The first word at addr must be a control word in the format shown previously. The length field must specify the number of 12-bit bytes occupied by the data in the PRU (maximum of 320 bytes, equivalent to 640 characters). The level number in the trailing control word specifies whether the PRU is an end-of-record or end-of-file. If a logical record extends over more than one PRU, the level number for the first PRU of the record must be nonzero and other than 17_8.

A leading and a trailing control word must be present for each PRU of data.

Typically, the data would consist of control statements which would comprise one record. Additional data to be processed by the batch job could also be present and typically constitute subsequent records.

Figure 5-4 demonstrates the SUBMT request.



length Length in 12-bit (two-character) bytes of job data following this control word (must be five times the number of central memory words).

level Logical record level number:

0 or omitted This physical record unit (PRU) is an end-of-record.

17_8 This PRU is an end-of-file.

Figure 5-3. SUBMT Parameter Table

```
00100  IDENTIFICATION DIVISION.
00110  PROGRAM-ID. SUBTEX1.
00120  ENVIRONMENT DIVISION.
00130  DATA DIVISION.
00140  COMMON-STORAGE SECTION.
                        .
                        .
Standard COBOL Communication Block  (refer to section 2)
                        .
                        .
00370  WORKING-STORAGE SECTION.
00380  01  JOB.
00390      03  FILLER              VALUE IS 1            COMP-4      PIC 9(5).
00400      03  FILLER              VALUE IS 0            COMP-4      PIC 9(5).
00410      03  JOB-LENGTH          VALUE IS 75           COMP-4      PIC 9(6).
       *
       *    THE FOLLOWING IS THE JOB TO BE SUBMITTED
       *
00420      03  JOB-CARD.
00430          05  JOB-ID          VALUE IS "JOB."                   PIC X(4).
00440          05  FILLER          VALUE IS ALL ":"                  PIC X(6).
00450      03  JOB-USER.
00460          05  USER-CARD       VALUE IS "USER,DWG2603,ABC123."   PIC X(20).
00470          05  FILLER          VALUE IS ALL ":"                  PIC X(10).
00480      03  JOB-CHARGE.
00490          05  CHG-CARD        VALUE IS "CHARGE,5927,693X451."   PIC X(20).
00500          05  FILLER          VALUE IS ALL ":"                  PIC X(10).
00510      03  JOB-DATA.
00520          05  COMMENT         VALUE IS "* SUBMIT BATCH JOB."    PIC X(19).
00530          05  FILLER          VALUE IS ALL ":"                  PIC X(11).
00540          05  CTR-STAT-1      VALUE IS "REWIND,INPUT."          PIC X(13).
00550          05  FILLER          VALUE IS ALL ":"                  PIC X(7).
00560          05  CTR-STAT-2      VALUE IS "COPYSBF."               PIC X(8).
00570          05  FILLER          VALUE IS ALL ":"                  PIC X(2).
00580          05  CTR-STAT-3      VALUE IS "ROUTE,OUTPUT."          PIC X(13).
00590          05  FILLER          VALUE IS ALL ":"                  PIC X(7).
       *
       *    SUBMITTED JOB ENDS HERE
       *
00600      03  INPUT-LEVEL         VALUE IS 15           COMP-4      PIC 99.
00610      03  FILLER              VALUE IS ALL ":"                  PIC X(8).
00620  PROCEDURE DIVISION.
00630  START-UP.
00640      ENTER SUBMT USING JOB.
00650      DISPLAY "JOB SUBMITTED.".
00660      STOP RUN.
```

Figure 5-4. SUBMT Request

The first word of JOB (00390-00410) sets bit 42 in the header word and gives the length (75) in 12-bit bytes of the card deck images to follow. This is 5 times the number of ten-character (60-bit) words in the job being submitted, in this case, statements 00420-00590.

Each job control statement must be from 1 to 80 characters long. All control statements to be submitted must be padded to a multiple of 10 characters and the last 2 characters must be binary zero (character ":" in the 64-character set). Thus, if a statement ends in character position 9,10; 19,20; and so on, a zero word must be appended. Statements 00460-00530 give examples of this.

A ROUTE statement (00580) is necessary to produce printed output.

A task can perform two types of dumps. The first type is a task dump, a dump of the task field length, exchange package, and associated data buffers. A task dump aids the applications programmer in debugging a task. The second type is a TAF dump, a dump of all or part of the TAF field length. A TAF dump contains pertinent information that is not contained in a task dump. This section describes the methods used to obtain both types of dumps.

TASK DUMP

The central memory dump (CMDUMP) request dumps central memory according to the parameters specified in the direct subsequent dumps (DSDUMP) request. Parameters specified on the DSDUMP request are used under three circumstances.

- When a fatal task error occurs.
- When a default value is requested in a CMDUMP request.
- When a parameter is in error in a CMDUMP request.

The purpose of DSDUMP and CMDUMP is to aid application programmers in debugging their tasks. These dump requests allow users to dump any portion of their programs. The user specifies the first word address (fwa) and the last word address (lwa) of the area to be dumped. The exchange package and/or all currently held TAF data manager data base file buffers may also be dumped by setting the proper parameter(s) to a nonzero value. With the CMDUMP command, users can specify particular TAF data manager file buffer areas in memory which are to be dumped.

The DSDUMP request sets up default values for subsequent CMDUMP requests made by this task and all linked tasks in this transaction (those called by CALLTSK without CEASE). DSDUMP does not cause a dump; it only sets parameter values that can be defaulted as a result of a CMDUMP request.

A hierarchy of parameter values allows users to default any parameters they want on DSDUMP or CMDUMP calls. At system assembly time, default values are assembled for each parameter. These default values can be changed via a K display command. In addition, if users want to change any default parameter values for their tasks, they may do so by using DSDUMP requests.

This hierarchy of default values works in the following manner. The CMDUMP request causes a dump according to the parameters it specifies. If a default option is requested, the system determines if a DSDUMP request has been made for this transaction. If a DSDUMP request has been made and if the corresponding parameter value has been supplied, this value is used. If DSDUMP requests have not been specified, a new default value, the system default value for this parameter, is used.

All task dumps, whether due to an aborted task, an abnormal CEASE request, or a CMDUMP request, are handled in the same manner. A job file is created containing the information to be dumped in binary format, as well as a control statement record. The transaction executive issues a ROUTE request releasing the job file to the input queue for batch processing. A program called KTSDMP is provided, either to generate listable output from the binary file or to transfer it to a direct access permanent file.

When dumping to an output queue, a ROUTE statement is included in the control statement record to direct the output produced by KTSDMP to the proper queue.

A USER statement is always included, thus enabling KTSDMP to attach the appropriate file if the dump is destined for a user's file. In this case, KTSDMP appends the binary information to the end of the permanent file. This method of handling dumps relieves the transaction executive of time-consuming processing that is better performed by a utility.

DSDUMP REQUEST

The DSDUMP request allows the application programmer to change any of the default values of the CMDUMP request. The DSDUMP does not cause a dump in itself, except when the system detects an error (that is, aborts). Any default values on the DSDUMP are satisfied from the general default values. All six parameters are required. If an error occurs on any DSDUMP parameter, the task is aborted.

The format is:

COBOL

ENTER DSDUMP USING fwa lwa ep db oq qd.

fwa — Computational-1 item which specifies beginning address of task memory to be dumped; legal values are≤0≤fwa FL (field length).

If fwa is negative, the default value is used.

lwa Computational-1 item which specifies last word of task memory to be dumped; it must be greater than fwa. If 0 is specified, no dump of task memory occurs. If lwa equals FL, the dump extends to the end of the field length. If lwa is negative, the default value is used.

ep Computational-1 item which specifies exchange package option.

- 0 Do not dump exchange package.
- > 0 Dump exchange package.
- < 0 Use default value.

db Computational-1 item which specifies TAF Data Manager buffer option (for use with TAF data manager only).

- 0 Do not dump TAF data manager buffers.
- > 0 Dump TAF data manager buffers.
- < 0 Use default value.

oq Computational-1 item which specifies output queue option.

- < 0 Use default value (see following note).
- 0 Dump to local batch output queue.
- 1 Dump to remote batch output queue.
- 2 Dump to user permanent file defined under data base user number.

qd Queue destination option; value depends on oq selection.

oq	qd
0	Printer ID, specified as a comp-1 number. If negative, use default value (see following note).
1 User number	A valid name specified in display-coded, left-justified characters with either blank or binary zero fill.
2 Direct access permanent file name	

NOTE

If either oq or qd is negative, the default value will be used for both, regardless of the option selected for the other.

FORTRAN

CALL DSDUMP(fwa,lwa,ep,db,oq,qd)

fwa Octal integer which specifies the beginning address of task memory to be dumped; legal values are 0≤fwa≤FL (field length).

If fwa is negative, the default value is used.

lwa Octal integer which specifies the last word of task memory to be dumped; must be greater than fwa. If 0 is specified, no dump of task memory occurs. If lwa equals FL, the dump extends to the end of the field length. If lwa is negative, the default value is used.

ep Integer which specifies the exchange package option.

- 0 Do not dump exchange package.
- > 0 Dump exchange package.
- < 0 Use default value.

db Integer which specifies TAF data manager buffer option (for use with TAF data manager only).

- 0 Do not dump TAF Data Manager buffers.
- > 0 Dump TAF Data Manager buffers.
- < 0 Use default value.

oq Output queue option.

- < 0 Use default value (see following note).
- 0 Dump to local batch output queue.
- 1 Dump to remote batch queue.
- 2 Dump to user permanent file defined under data base user number.

qd Queue destination option; value depends on oq selection.

oq	qd
0	Integer specifying printer ID. If negative, use default value (see following note).
1 User number	A valid name specified in display-coded, left-justified characters with either blank or binary zero fill.
2 Direct access permanent file name	

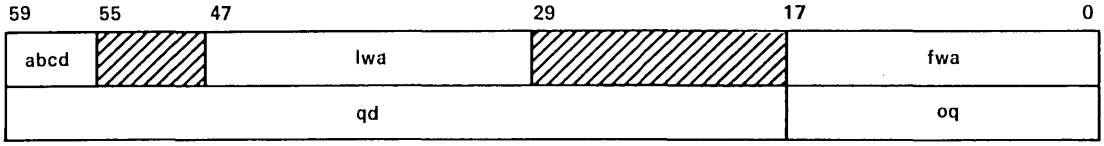
NOTE

If either oq or qd is negative, the default value will be used for both, regardless of the option selected for the other.

COMPASS

DSDUMP addr

addr Address of a parameter table. The table has the form shown in figure 6-1.



abcd Bits 59 through 56 of first COMPASS parameter word.

59 Set to 1 for exchange package dump.

58 Set to 1 for TAF data manager buffer dump.

57 Set to 1 for exchange package default value.

56 Set to 1 for TAF data manager buffer default value.

lwa Last word of task memory to be dumped; must be greater than fwa. If 0 is specified, no dump of task memory occurs. If lwa equals FL, the dump extends to the end of the field length. If lwa is negative, the default value is used.

fwa Beginning address of task memory to be dumped; legal values are 0 fwa FL (field length).

If fwa is negative, the default value is used.†

qd Queue destination option; value depends on oq selection.

oq		qd
0	Printer ID, specified as a binary number right-justified within word; if negative, use default value.	
1	User number	A valid name specified in display-coded, left-justified characters with either blank or binary zero fill.
2	Direct access permanent file name	

oq Output queue option.

0 Dump to local batch output queue.

1 Dump to remote batch output queue.

2 Dump to user permanent file defined under data base user number.

†Default values are specified by setting the sign bit.

Figure 6-1. DSDUMP Parameter Table

CMDUMP REQUEST

The CMDUMP request allows the application programmer to dump memory in the task's field length and/or data manager buffers to a selected output queue (and logical device). The first six parameters are required. If there is an error on or a default of any parameter specified by CMDUMP, the parameter value of the DSDUMP call is used.

The format is:

COBOL

ENTER CMDUMP USING fwa lwa ep db oq qd $file_1$ $file_2$,...,$file_n$.

The descriptions of the first six parameters are the same as those for the DSDUMP request.

$file_i$ File name of a data manager buffer to be dumped. Legal values for $file_i$ are four display-code characters, left-justified. If no file names are present, a dump of all data manager buffers is implied.

FORTRAN

CALL CMDUMP(fwa,lwa,ep,db,oq,qd,$file_1$, $file_2$,...,$file_n$)

The descriptions of the first six parameters are the same as those for the DSDUMP request.

$file_i$ File name of a data manager buffer to be dumped. Legal values for $file_i$ are four display-code characters, left-justified. If no file names are present, a dump of all data manager buffers is implied.

COMPASS

CMDUMP addr

addr Address of a parameter table in the form shown in figure 6-2.

NOTE

The manner in which the hierarchy of default values works was previously described. However, in the special case of output queue and queue destination (oq and qd parameters), if either parameter is defaulted, the next pair of output queue and queue destination parameters is obtained from the next level of the default hierarchy.

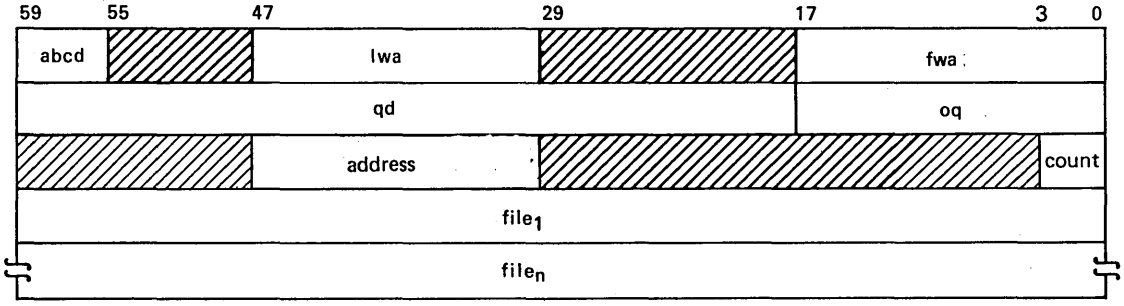
There also is a special case concerning errors in a CMDUMP request. If the parameter in error is the output queue value, the system is unable to route the dump as the user wants. If this error is on a CMDUMP request, the system attempts to dump according to the corresponding DSDUMP output queue and queue destination parameters. If that is not possible, the general system values are used.

K.DSDUMP COMMAND

The K.DSDUMP command can be used under the K display for the transaction executive control point to reset default dump parameters. The parameters can be given in any order, and only those parameters to be changed need to be given. If the data base option is specified, all TAF data manager file buffers currently belonging to a transaction are dumped when a dump defaults to this parameter. If (DB)=0, no buffers are dumped.

The format when entered from the console is:

K.DSDUMP,FW=fwa,LW=lwa,EP=ep,OQ=oq, QD=qd,DB=db.



The descriptions of the first five parameters are the same as those shown in figure 5-1.

address User's calling address to be placed on listing.

count Number of file names present.

$file_i$ File name of a TAF data manager buffer to be dumped. Legal values for $file_i$ are four display-code characters left-justified and binary zero filled. If no file names are present, a dump of all data manager buffers is implied.

Figure 6-2. CMDUMP Parameter Table

The values of these parameters are as specified for the DSDUMP request made by a task. These system dump parameters can be examined by checking the K display for control points.

KTSDMP UTILITY

KTSDMP can be called via a control statement to form listable memory dumps. The input files can be created by the transaction executive or by any other program. The following items can be listed.

- Central memory of a task or program
- Exchange package and control point area of a task
- Communication block of a task (always present)
- TAF data manager data buffers

The format is:

KTSDMP (lfn_1,lfn_2,P,O)

lfn_1 Input file name; if absent, default is INPUT.

lfn_2 Output file name; if absent, default is OUTPUT.

P If P is specified, append file lfn_1 on direct-access permanent file lfn_2. The job must be running with the proper user number, and file lfn_2 must be defined. End-of-file is not copied to lfn_2.

O If O is specified, the dump is in octal format. If O is omitted, the dump is in octal and display code format, with the display code format to the right of the octal format.

If the P option is used, an installation may keep a running stack of memory dumps to be listed selectively at some later time. To list all dumps on the permanent dump file, the file should be attached and KTSDMP executed. All dump records on the file are processed. To list only selected dumps, the permanent file should be attached and the selected dump records extracted using GTR (refer to the NOS Reference Manual, volume 1). KTSDMP should be executed to process the output file from the GTR request.

The user must specify the O option when issuing the KTSDMP control statement from a terminal; if the O option is not specified, execution is terminated, and KTSDMP issues the following message.

DISPLAY DUMP NOT ALLOWED TO TERMINAL.

MEMORY DUMP FORMATS

Figure 6-3 contains the formats of the dumps provided. The leftmost date and time in the header are the date and time of creation of the KTSDMP input file. The current date and time follow. The task control point area dump appears only when the exchange package dump is requested. The exchange package dump, task control point area dump, and the communication block dump all appear on the same page when the exchange package dump is requested.

TAF DUMP

A TAF dump is a dump of all or part of the TAF field length. The user initiates a TAF dump using the K.DUMP command. A TAF dump assists the analyst in detecting system errors which do not abort TAF. When a task detects an error in TAF, a TAF dump might be needed because there is significant information outside the task field length that is not contained in a task dump.

A task initiates a TAF dump using the KPOINT request (refer to KPOINT request in section 2) in conjunction with the K.DUMP command. The K.DUMPLIM command sets a global limit on the number of K.DUMP commands that tasks can issue. A task uses the KPOINT request in conjunction with the K.DUMPLIM command to set the global task dump limit (GTDL). The remainder of this section describes the K.DUMP and K.DUMPLIM commands.

K.DUMP COMMAND

The K.DUMP command initiates a dump of all or part of TAF. The user can specify the first word address (fwa) and the last word address (lwa) of the memory to be dumped.

The command formats when issued from a task are:

DUMP,fwa,lwa.
or
DUMP,lwa.
or
DUMP.

fwa Octal address of the first word of memory to be dumped; legal values are $0 \le fwa \le lwa$. The default is 0. This parameter cannot be used unless the lwa parameter is also used.

lwa Octal address of the last word of memory to be dumped; legal values are $fwa \le lwa \le 377777$. The default is 377777.

If both parameters are omitted, the entire field length is dumped. The output of the dump is routed to a printer which has an identifier of zero. A header page precedes the dump; it lists the date and time of the dump and the name of the task that issued the dump. Also listed are the instructions that the dump be given to the TAF central site systems analyst.

K.DUMPLIM COMMAND

The K.DUMPLIM command sets the GTDL. The GTDL specifies the number of K.DUMP commands that tasks can issue.

```
KTSDMP - tasknam                    yy/mm/dd. hh.mm.ss.          yy/mm/dd.hh.mm.ss.
EXCHANGE PACKAGE.
    Exchange package dump
TASK CONTROL POINT AREA.
    Task control point area dump
COMMUNICATION BLOCK DUMP FROM fwa TO lwa
    PROGRAM tasknam   SEQUENCE num   ADDRESS xxxxx
    Communication block dump
KTSDMP - tasknam                    yy/mm/dd. hh.mm.ss.          yy/mm/dd. hh.mm.ss.
DUMP FROM fwa TO lwa
    Memory dump
KTSDMP - tasknam                    yy/mm/dd. hh.mm.ss.          yy/mm/dd. hh.mm.ss.
DUMP FROM fwa TO lwa
    Memory dump
KTSDMP - tasknam                    yy/mm/dd. hh.mm.ss.          yy/mm/dd. hh.mm.ss.
DATA BUFFER DUMP OF db filen
KEY VALUE
    Value
PRU         xxx
STATUS      xxx
    File buffer dump
```

Figure 6-3. KTSDMP Format

The command format when issued from a task is:

DUMPLIM,n.

n The decimal number of TAF dumps that tasks can issue. The default is 0. The maximum value is 9999999. The user specifies octal by using a postradix of B.

The initial value for the GTDL is zero. When the GTDL is zero, no TAF dumps from tasks are allowed. The GTDL must be set to a value greater than zero for TAF dumps from tasks to occur. Each TAF dump (where the task does not abort) causes the GTDL to be decreased by 1. This command does not affect the execution of task dumps.

This section can be omitted by the applications programmer.

RECOVERY

Transaction control-point recovery occurs under the following conditions when sense switch 4 is set.

- If the transaction subsystem is aborted for any reason.
- If the entire operating system is recovered (level 3 system deadstart recovery).

The sense switch is set by the operator. (Refer to the n.ONSWx command in the NOS Operator's Guide.) If automatic recovery is not desired, sense switch 4 should not be set.

Recovery is automatic, requiring no operator interaction. The following dayfile messages are issued during recovery.

```
RECOVERY IN PROGRESS.
n.nnn KILO TRANSACTIONS PROCESSED.
n.nnn KILO TRANSACTION ABORTS.
n.nnn KILO TASK RELOADS.
n.nnn KILO ITASK RELOADS.
n.nnn KILO TAF FL INCREASES.
n.nnn KILO STORAGE MOVES OF TASKS.
n.nnn KILO NO FL FOR TASK LOAD.
n.nnn KILO NO SUBCONTROL POINTS.
n.nnn KILO NO COMMUNICATION BLOCKS.
n.nnn KILO TASK ROLLOUT STARTS.
n.nnn KILO TASK ROLLOUT COMPLETES.
n.nnn KILO TASK ROLLOUTS BY TAF DM.
n.nnn KILO TASK STARTS BY TAF DM.
n.nnn KILO TASK RECALLS.
n.nnn AVERAGE ACTIVE SUBCONTROL POINTS.
n.nnn AVERAGE OUTSTANDING CDCS REQUESTS.
n.nnn KILO CDCS REQUEST REJECTS FOR MAXR.
n.nnn KILO CDCS REQUEST REJECTS FOR BUSY.
n.nnn KILO CDCS REQUESTS FROM TASKS.
n.nnn PER CENT CPU USAGE.
RECOVERY COMPLETE.
```

The following dayfile messages are issued during recovery only if the CRM data manager is being used.

```
n.nnn KILO OPENS - filenam.
n.nnn KILO OPEN REJECTS - filenam.
n.nnn KILO LOCKS - filenam.
n.nnn KILO LOCK REJECTS - filenam.
AAM FILES CLOSED.
```

The filenam field is the name of the CRM permanent file.

The transaction abort count is the number of times a transaction did not end normally due to a transaction-executive-detected error in a task while processing the transaction. Errors of this type include mode errors (in the task), illegal parameters on a request from a task, and task CPU time limit exceeded (possible infinite loop in task). This count does not include transactions terminated abnormally by use of the CEASE request.

If sense switch 4 was not set, the preceding messages are still recorded but the transaction executive TERMINATE message is issued instead of the RECOVERY COMPLETE message. If sense switch 5 is set, the transaction executive dumps its field length and releases the output file to the local print queue after each fatal error or operator drop.

If recovery is attempted before initialization is complete, the following messages appear.

```
RECOVERY IN PROGRESS.
RECOVERY IMPOSSIBLE.
```

NOTE

The procedure file used to initialize TAF must include the proper control statements to recognize sense switches and perform dumps and recovery. Refer to the NOS Installation Handbook for further information.

The first action taken during recovery is to flush buffered journal files. The initialization overlay is then called to reload the transaction subsystem data manager code and build memory resident tables, such as the terminal status table and task library directories. After the tables are constructed, the recovery flag (word 0 of the communication block) is set in the user area of each terminal status table entry to enable initial task or any other task to detect that a recovery has taken place. The location of the flag in the area may be established by each installation. The flag is accessible by the use of either the TARO or TSIM request. The flag can be turned off by the TARO request.

The rebuilding of memory resident tables nullifies any changes made by K display commands prior to the recovery. It is the responsibility of the operations personnel, after a recovery has occurred, to enter desired changes, if any. Also nullified are any changes made to the user areas of the terminal status table by the TARO request. It is the responsibility of the applications programmer to reset or clear the bits that were changed prior to recovery.

No transactions are recovered; thus, it is the user's (application task's) responsibility to recover from partially completed transactions and their effects on the data base.

A K display command K.REC=YES. can be used to set terminal status table recovery bits during a normal initialization. This could be used after a 0 level deadstart to force the terminals into recovery mode.

Under the following conditions, the transaction executive sets the recovery bit in the user area of the terminal status table.

- If the K display command K.REC=YES. is entered at initialization time, the transaction executive sets the recovery bit for all terminals.
- If the time-sharing executive aborts while the transaction executive is running, the transaction executive clears the log-in bit and sets the recovery bit for all terminals currently logged into the transaction subsystem. If any tasks are rolled out waiting for terminal input, the transaction executive sets their rollout time delays to zero such that they are rolled in.
- If a transaction terminal user hangs up the phone or a line problem occurs, the transaction executive clears the log-in bit and sets the recovery bit for that terminal name. Clearing the log-in bit enables the terminal user to reenter transaction mode.

Any of these conditions which cause the transaction executive to set the recovery bit indicates a possible loss of transaction terminal output. The application task monitors the recovery bit in the user area field and takes appropriate terminal/application resynchronization procedures when the bit is set.

OPERATOR INTERFACE

A console K display is provided to display transaction subsystem status at all times. A set of console operator commands is also available for changing certain subsystem parameters. The NOS Operator's Guide supplies a description of the commands (refer to preface for publication number).

The applications programmer should determine whether an analyst is preparing the network files. If so, this section can be omitted.

Before any transaction processing can be done, the terminal configuration must be defined to the transaction executive. This is done by the terminal network description file (network file).

The transaction subsystem uses multiplexers to provide communication between the terminals and the PPUs. Only 6671 (16 ports) and 6676 (64 ports) are supported.†

TERMINAL NETWORK DESCRIPTION FILE

A terminal network description file consists of directives that describe the configuration.

The file contains one directive for each multiplexer in the network, with each multiplexer directive followed by a set of terminal directives for the set of terminals communicating through that multiplexer.

A multiplexer directive has two possible forms. They are:

```
//muxtype,CH=n1,EQ=n2.
        and
//DIAL.
```

The muxtype parameter indicates the model designation of a multiplexer (6671 or 6676). The CH parameter gives the channel number, with n1 ranging from 0 to 13B and 20B to 33B, designating the channel through which the multiplexer is connected to the PPU. The EQ parameter gives the system equipment number, with n2 ranging from 0 to 7. These parameters are required where shown.

The //muxtype form results in a permanent association of a multiplexer with a set of terminals.

The //DIAL form does not refer to any particular multiplexer or port. It requests that all terminals specified under it be assigned to whatever multiplexer ports are available when such assignment is needed. In the latter equipment assignment, no one multiplexer port is permanently restricted to a fixed set of equipment.

Terminal directives have the following format.

```
/terminal, parameter-string.
```

Terminal is a unique alphanumeric name (one to seven characters) that identifies the terminal or user number. The terminal name is required.

The possible elements of the parameter string (separated by commas) are:

PN=n1 Port number n1 of the multiplexer with which the terminal is associated. It is required only if the terminal is associated with a specific multiplexer equipment; that is, if the preceding multiplexer directive was a //6671 or //6676 entry. The port number n1 ranges from 1 through 16 for a 6671 multiplexer, and from 1 through 64 for a 6676 multiplexer. All other parameters are optional.

PL=n2 Port range limit which is meaningful only when the PN parameter is used. It restricts the range of ports that can be used with one multiplexer, and it can do this only after the port number has been established. For example, the terminal definition

```
/TERMX, PN=2, PL=5
```

connects terminal TERMX to port 2 and specifies that only five more ports (ports 3 through 7) can be given terminal definitions.

TT=*aaa Transaction terminal type. *aaa can be either of the following:

*AB Terminal identified by the answerback feature.

*ID Terminal identified by terminal operator entry.

If the terminal is to be declared for nontransaction use, the * is omitted from the parameter.

LS=n Line speed, that is, the number of characters per second (cps) that can be transferred to or from the terminal. n can be any legal cps rate.

DB=aa Two-character name of the data base to be accessed from the terminal. Only one data base can be associated with a terminal name. This parameter is required in a directive that names a transaction-only terminal. Refer to *aaa parameter previously.

IS=aaa Initial ON/OFF status of the terminal. Valid values are ON and OFF, with ON the default.

RS=n The data manager read security for the terminal, with respect to the data base, is specified by the RS parameter. The terminal cannot access data elements whose DATADEF read security is higher than its own. n ranges from 0 to 7, with 7 the most exclusive. Default value is 0.

†There is one exception; a 2550-1 simulates a 667x multiplexer and is, therefore, treated the same as a multiplexer.

US=n This declares the data manager update security for the terminal. The terminal cannot update elements with update security higher than its own. n ranges from 0 to 7, with 7 the most exclusive. Default value is 0.

UA=n Contents of the user area in the communication block. n ranges from 0 to 16777215. Default value is 0. The contents of the user area can be examined and changed by tasks. The appropriate initial value is site dependent. Bit zero is the default task recovery bit.

The following is a simple set of terminal network file directives.

```
//6671,CH=2,EQ=3.
/TTY1,PN=4,TT=*AB,DB=BC,RS=7,US=7.

//DIAL.
/AJOHNSN,TT=*ID,RS=7,US=3,DB=BC,
 UA=77117711B.
/USERNUM,TT=*ID,RS=6,US=7,DB=BC,
 UA=00007700B.
/XYZ1,TT=*ID,RS=6,US=5,DB=BC.
```

This set of directives can be entered on cards or from the system origin.

VALNET VALIDATION

Validate network (VALNET) is a system program that checks the syntax and logic of a terminal network description file.

The format is:

```
VALNET(p1,p2,p3)
```

The p parameters can be in any order and are:

P Specifies the name under which the terminal network file is stored. Options are:

- P — File name is COMPILE.
- P=fn — File name fn is supplied by user.
- Omitted — Default file name is NETWid where id is the two-character machine identifier. This can be obtained by using the MACHID macro which is described in the NOS Reference Manual, volume 2 (refer to preface for publication number).

L Specifies the file that receives the list of errors. Options are:

- L — File name is LIST.
- L=fn — File name fn is supplied by user.
- L=0 — No list is produced.
- Omitted — Default name is OUTPUT.

NR Specifies whether the network description should be rewound before VALNET activity commences.

- NR — No rewind before VALNET reads the network description file.
- Omitted — Rewind before reading.

For each error encountered in a network description file, VALNET produces two output lines with the following format.

Error Line	
Statement Number	Diagnostic Message

The error line is the faulty line, or in the case of an omission, a near subsequent line. As an example, the descriptive code

```
(line 2)    //6676,CH=5,EQ=6.

(line 3)    /TERMA.

(line 4)    /TERMB,PN=3.
```

is missing the required port number in line 3. However, this is not diagnosed until line 4 is encountered. Line 4 is listed along with the appropriate diagnostic as follows:

```
TERMB,PN=3
CARD 04 PORT NUMBER UNDEFINED FOR
PREVIOUS TERMINAL.
```

Output is produced only if errors are encountered. Error messages are described in appendix B.

NETWORK FILE ORIGINATION

When an error-free network description is ready, an analyst should create a system file called NETWid. Refer to the P parameter definition under VALNET Validation.

A typical procedure for originating an NETWid file follows. First, the user establishes a set of network description statements as a local file.

```
NETW.
USER(usernumber,password,familyname)
CHARGE(chargenumber,projectnumber)
COPYBR(INPUT,NET)
SAVE(NET)
7/8/9
     Network description statements
6/7/8/9
```

Then the user enters the following at the system console.

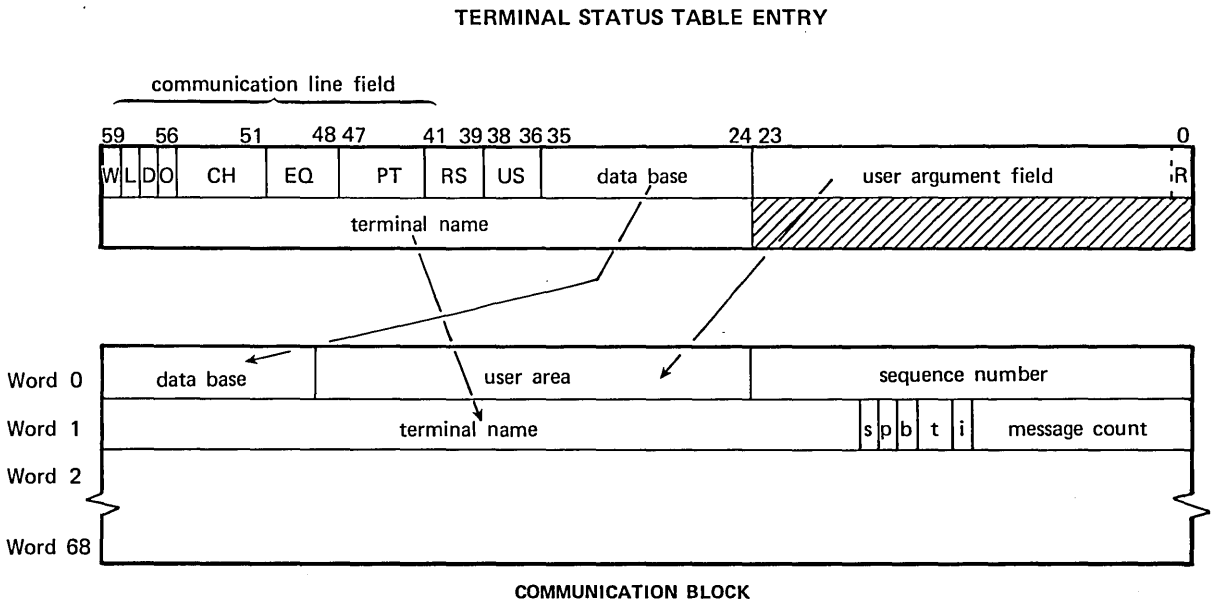
```
X.DIS.
USER(usernumber,password,familyname)
CHARGE(chargenumber,projectnumber)
GET(NET)
SUI(377777)
DEFINE(NETWid/CT=PU)
COPY(NET,NETWid,V)
RETURN(NETWid,NET)
```

This creates a direct access public file called NETWid under the system user index 377777. NETWid is used to generate terminal status table (TST) entries.

Figure 8-1 illustrates a terminal status table entry. Each line gives the terminal name and its associated data base, read and update securities, and user area value.

When a transaction originates from a terminal, the transaction executive uses the appropriate entry in the terminal status table to construct words 0 and 1 of the communication block.

After the network file is originated, it must be updated to reflect changes, and the time-sharing and transaction executive must be reinitialized to recognize the new information.



ID	Function
W	Interactive task waiting for input
L	Terminal user active
D	Terminal down
O	Terminal on/off
CH	Multiplexer channel
EQ	Multiplexer equipment
PT	Multiplexer port
RS	Data base read security level
US	Data base update security level
R	Default recovery bit

Figure 8-1. Communication Between Terminal Status Table and Communication Block

A task library contains all of the tasks which may be requested by a user group within the transaction subsystem. The task library is created by the LIBTASK utility.

A task library file is a direct-access permanent file with no password that must be created with a DEFINE command before a library creation run occurs. However, it does not have to be attached by the user unless the DA option is specified (refer to LIBTASK control statement). The transaction subsystem simultaneously supports one system task library, and optionally, one task library for each data base.

The system task library, called TASKLIB by default, may be renamed at the user's option. It is defined under the transaction subsystem user number and given write permission to that user number. The name TASKLIB may also be changed using the K display command K.TLF. (Refer to the NOS Operator's Guide.)

A user task library must be named xxTASKL where xx is the name of the data base with which the library is associated. It must be defined under the data base user number, and explicit write permission must be granted to the transaction subsystem user number to access the file.

During initialization, an attempt is made to attach a library for each initialized data base. The xxJ file is accessed to determine the user number needed to attach the library. When a scheduling request is received, the transaction subsystem searches for the task first in the associated data base library, if one exists, and then in the system library. Each user can, therefore, have tasks with the same name, and task security between tasks is still maintained. The only exception to this procedure is the initial task, which is always loaded from the system library.

LIBTASK UTILITY

The LIBTASK utility creates a new library, adds new tasks to an existing library, replaces tasks on an existing library, and compacts an existing library by removing inactive records. Tasks may be added or replaced while the transaction subsystem is running, because tasks are added as a new file at the end of the library. The library is a multifile file. The transaction executive attempts to read a new library directory if the LIBTASK control statement contains the TT option. Thus, on-line update of any task library is allowed.

If the library is modified while the transaction subsystem is running and the user wants the new directory used, the user number and password of the LIBTASK user are passed to the transaction subsystem for validation. This is done by intercontrol point communication, which is also used to check if the transaction subsystem is running if the CR or PR option is selected. Therefore, the first USER statement in the job must have permission to do intercontrol point communication. (Refer to the NOS System Maintenance Reference Manual for the use of MODVAL to alter permissions.) The LIBTASK run issues an error message and does not update the library if the validation fails. The task binaries to be added or replaced are in ABS or OVL format produced by the CYBER Loader.

TASK RESIDENCY AND TYPE

An important option of LIBTASK enables the user to specify the residency and type of tasks on the task library. A task can reside on central memory, ECS, or mass storage. This can be specified with the C and E options on LIBTASK directives.

A task can be of either reusable or nonreusable type. A reusable task must be written so that it is reinitializing. A reinitializing task (program) does not modify program areas that the program considers constant, or if it does, it resets (reinitializes) these areas upon reentry. Multiple requests for a reusable task can be queued together; this is partly under the control of the person who sets up the task library and partly under the control of the transaction executive. The queuing of multiple requests occurs in an internal list, the Requested Task List (RTL). Each entry in this list represents a scheduling request for a copy of a task. There may be multiple entries in the RTL list for a given task. The LIBTASK queue limit input directive can be used to specify the maximum number of transactions which can be queued against one entry. When the transaction executive schedules a copy of a task from the RTL list, it picks an RTL entry and forces all of the multiple requests queued for that RTL entry to use the same copy of the task.

It is generally possible to have multiple copies of a given task in execution at the same time (Note that this does not apply to CM resident tasks). Additional application control of task scheduling is provided by the Q LIBTASK option in the input directives.

A nonreusable task is not a reinitializing task; a new copy of the task must be loaded from the task library for each transaction.

The task type can be specified with the D option on LIBTASK directives. The queue limit can be specified with the QL option.

If a task is specified as central memory resident and reusable, it is loaded to a subcontrol point when TAF is initialized and, unless the task aborts, remains at that subcontrol point to service transactions as long as TAF is running. If the task aborts, another copy of it will be loaded. It is a single copy task. No additional copies of this task are loaded to process requests after queue limit has been reached.

LIBTASK CONTROL STATEMENT

The LIBTASK control statement has two formats. The first format allows specification of the file containing input directive statements; this format is:

LIBTASK(p_1,p_2,...,p_n)

The second format specifies that input directive statements are included in this LIBTASK control statement; this format is:

LIBTASK(p_1,p_2,...,Z,...,p_n)/d_1/d_2/.../d_n.

/ Unique delimiter that does not occur in any d_i input directives. This delimiter does not have to be a slash.

d_i Input directive statement (refer to Input Directives in this section).

The input directives on a LIBTASK statement remain in effect for a task in an existing task library until the task is specified in a subsequent LIBTASK statement; that is, the most recent input directives (including default input directives) for a task take precedence over earlier input directives. However, this is not true for a task that is not in a task library; in this case, the first input directives (including default input directives) take precedence.

For example, the user specifies the following statement:

LIBTASK(p_1,p_2,p_3,...,p_n)/*T1 DL.

This statement logically deletes task T1. If the user specifies task T1 in a subsequent LIBTASK statement and does not specify DL, the task is activated (by default).

The LIBTASK control statement cannot exceed 80 characters.

The p parameters for either format can be in any order and are:

Parameter	Meaning
I	Specifies the file containing input directives. The file is not rewound before processing. All directives that do not refer to a task name present on the B file or P file are ignored. Options are:
	I — File is COMPILE.
	I=lfn — File is specified by user.
	I=0 — No directive file exists.
	Omitted — Default file is INPUT.
B	Specifies the file containing task binaries to be added to the library. These may be tasks already existing in the library or they may be new ones. The file is rewound before processing. Options are:
	B — File is LGO.
	B=lfn — File is specified by user.
	B=0 — Normal output information is given.
	Omitted — Default file is LGOB.
L	Specifies the file on which output is written.
	L — File is OUTPUT.
	L=lfn — File is specified by user.
	L=0 — No output is written.
	Omitted — Default file is OUTPUT.
P	Specifies the name of a direct-access permanent file to be used as the library file. The file is required. Options are:
	P — File is TASKLIB.
	P=lfn — File is specified by user.
	P=0 — No old library file exists.
	Omitted — Default file is TASKLIB.
N	Specifies a newly defined direct-access permanent file to be written as the new library file. Use of this option eliminates the complete copy of the P file to a scratch file on the PR option, thus significantly reducing the purge time for long libraries. The P file is not purged. Options are:
	N — File is TASKLIB.
	N=lfn — File is specified by user.
	Omitted — Default file is the file specified by P.
DA	Disables all attaches and returns of the library files in the LIBTASK program. With this option selected, the N file and the file specified by P must be attached prior to the LIBTASK call. If a library is attached in the proper mode from another user name, the DA option must be selected so that LIBTASK does not try to attach the appropriate libraries under the user name from which the update is being run.
CR	Creates a new task library. LIBTASK reads one task at a time from the B file, checks for an input directive for the task, constructs a directory entry, and inserts the entry in alphabetical order in the library directory. The directory is written at the end of the library. This option applies only if LIBTASK was able to attach the task library (specified by the N option) in write mode.
PR	Removes inactive task binaries and old directories from the task library. LIBTASK processes the B file and the input directives normally, and if N is not specified, copies the entire library to a scratch file. The current binary for each task in the

Parameter	Meaning
	library is then transferred from the scratch file to the library file, and the directory is updated and written at the end. Therefore, only one copy of each task and one directory remain in the library. If N is specified, no scratch file is written, and P is not purged. The new library file contains only the latest copy of each active task and one directory. This option applies only if LIBTASK was able to attach the task library (specified by the N option) in write mode.
TT	Instructs the transaction executive to update the TAF resident copy of the task library directory while TAF is running. If not specified, TAF will not become aware of the update until the next time it is initialized. The installation parameter TLDL is used to specify the amount of space, in words, to be reserved for additions to each TAF resident task library directory. The default value is 30 which allows 10 additions (three words for each TLD entry). Although further additions can be made to the user's task library, they will not be recognized by TAF until TAF is reinitialized. There is no limit to the number of times existing tasks can be replaced. The user number from which LIBTASK is being run must be the same as the user number in the xxJ file associated with the library being modified, and TAF must be able to attach the library in WRITE mode in order for the update of TAF's copy of the TLD to take place. The user must be validated for intercontrol point communication (the CSTP bit in user's access word must be set) to use the TT option.
Z	Specifies that input directives are on the LIBTASK statement. This option overrides the I option.

INPUT DIRECTIVES

To alter task characteristics, directive statements may be entered in the directive file. Only one statement should appear for each task. These characteristics are kept in the directory for use by the transaction executive when the task is to be executed.

The directive statement format is:

*name $p_1,p_2,\ldots,p_n$.

name is the task name and p_i is any of the following input directives in any order.

Directive	Meaning
BP=nn	Beginning scheduling priority of task.
BP	Beginning priority defaults to 25B.
BP omitted	Beginning priority defaults to 20B.
C	Specifies a memory-resident task. The default is a disk-resident task.
D	Specifies a destructive code (nonreusable) task that is removed from the subcontrol point when the task is completed. The default is a reusable task. This option is recommended when debugging a task.
DL	Specifies that a task (name) is logically deleted. All logically deleted tasks are physically deleted from the task library when the LIBTASK statement contains the PR option.
E	Specifies an ECS-resident library copy of the task. The default is mass storage resident. If tasks are specified as ECS-resident, the transaction executive during initialization loads as many as will fit into its available ECS and leaves the remaining tasks on the mass storage file. If no ECS is available, all tasks are loaded from mass storage. If a LIBTASK function specifying the TT option is issued while the transaction executive is running, no tasks are replaced or added to ECS, and one of the following situations occurs. • Old copy of task in ECS and new copy in ECS. The new copy is ignored, and the old copy remains on the library in ECS. • Old copy not in ECS and new copy in ECS. The new copy is used but is read from mass storage. • Old copy in ECS and new copy not in ECS. The new copy is loaded from mass storage.
EF=efl	Specifies the maximum amount by which a task will be allowed to increase its initial field length. efl is assumed to be octal unless it contains an 8 or 9 or is followed by a trailing D. The default value is zero. EF will be increased to the next multiple of 100_8 if it is not already such a multiple. The maximum field length (MFL) allowed for the task is (initial task FL) + efl. The value of efl can range from 0 to 300000_8.
IG	Specifies that a task (name) in the file specified with the B option is ignored.
MP=nn	Maximum scheduling priority of task (for future use in aging tasks).
MP	Maximum priority defaults to 50B.
MP omitted	Maximum priority defaults to 40B.
O	Sets task to OFF; requests for the task are routed to the OFFTASK routine (refer to OFFTASK in section 10). The default is ON.

Directive	Meaning
Q	Specifies that queuing is to be forced as described below. This option alters the normal scheduling priority of the task (refer to Scheduling Priority) as follows: 1. When there are no active copies of the task, the first copy of the task is scheduled as usual. 2. While there are from one to six active copies of the task in TAF, another copy of the task is not scheduled until the number of transactions queued against its RTL entries equals or exceeds the queue limit specified by the QL directive. When an RTL entry is used to schedule a task, it is released and made available for requesting the scheduling of other tasks. 3. When the number of active copies of the task equals 7, additional copies of the task will not be scheduled. This applies even if the queue limit is 1. An active task is a copy of a task which is: a) at a subcontrol point or rolled out, and b) in the process of handling a transaction or has transactions queued against it (the copy). If a LIBTASK function specifying the TT control statement option is issued while the transaction executive is running and an attempt is made to change the Q-attribute of a task on a library in use by TAF, then the Q-attribute change will not affect TAF scheduling until the executive is reinitialized. This parameter is not applicable to CM resident tasks.
QL=n	Specifies the number of requests, up to a maximum of five, that can be queued for an RTL entry for a task. If a task uses a large amount of memory and executes rapidly, performance is not degraded if requests are queued. In this case, multiple copies of the task require too much memory. If a task does a large amount of I/O, then performance is degraded if requests are queued. In this case, multiple copies of the task enhance performance.
QL	Queue entry limit defaults to five.
QL omitted	Queue entry limit defaults to three.
R	Specifies that the FL assigned to a CM resident task is to be reduced to the initial load value each time the transaction is run. The default is no reduce. Non-CM resident tasks are always started with their initial load value.
SC	Sets the task communication block load status to solicited. This means the user must request the communication block and supply a valid address (using the BEGIN request) after the task is initiated. The default is unsolicited.

All directives for one task must appear on the same line. The following situation will result in only p_1 being recognized as an input directive for TASK1.

```
*TASK1 p1.
*TASK1 p2.
   .          Improper way to specify
   .          input directives.
   .
*TASK1 pn.
```

SCHEDULING PRIORITY

When the tasks are queued, the scheduler will attempt to load the task with the highest current priority. If there are several tasks with the highest priority, the task with the lowest field length requirement will be selected for loading. When a task enters the queue, the beginning priority (BP) is used as the task's current priority. The current priority is then aged (incremented by one) every second up to the maximun priority (MP). Tasks in the queue having identical current priorities and field length requirements normally will be scheduled in the order they entered the queue.

With these directives, classes of tasks can be created. For example, if two classes were created (X and Y) their priorities could be:

$X_{BP} = 25_8$ $\quad X_{MP} = 40_8$
$Y_{BP} = 30_8$ $\quad Y_{MP} = 45_8$

In this case, no tasks in X could be scheduled until all tasks in Y having a current priority greater than 40_8 had been scheduled.

CREATING A NEW TASK LIBRARY

The following examples show how a user can create a new task library. An explanation is provided after each LIBTASK statement.

```
LIBTASK(P=0,N=lfn1,CR,B=lfn2,I)
```

LIBTASK copies file lfn2 to file lfn1 according to input directives in the COMPILE file and adds a new sorted task library directory at the end of file lfn1.

```
LIBTASK(P=lfn2,N=lfn1,Z)/*name p1,p2,...,pn.
```

LIBTASK copies file lfn2 to file lfn1 and adds a new sorted task library directory at the end of file lfn1 according to input directives on the LIBTASK statement.

```
LIBTASK(P=lfn2,N=lfn1,PR,Z)/*name p1,p2,...,pn.
```

LIBTASK copies all active (not logically deleted or replaced) task binaries from file lfn2 to file lfn1 and adds a new sorted task library directory at the end of file lfn1 according to input directives on the LIBTASK statement.

```
LIBTASK(P=lfn2,N=lfn1,B=lfn3,I)
```

LIBTASK copies file lfn2 to file lfn1, copies file lfn3 to file lfn1 according to input directives in the COMPILE file, and adds a new sorted task library directory at the end of file lfn1.

```
LIBTASK(P=lfn2,N=lfn1,B=lfn3,PR,I)
```

LIBTASK copies all active (not logically deleted or replaced) task binaries from file lfn2 to file lfn1, copies file lfn3 to file lfn1 according to input directives in the COMPILE file, and adds a new sorted task library directory at the end of file lfn1.

UPDATING AN EXISTING TASK LIBRARY

The following examples show how a user can update (add new tasks and delete and replace existing tasks) an existing task library without supplying any binaries. An explanation is provided after each LIBTASK statement.

```
LIBTASK(P=lfn1,Z)/*name p1,p2,...,pn.
```

LIBTASK adds a new sorted task library directory at the end of file lfn1 according to input directives on the LIBTASK statement.

```
LIBTASK(P=lfn1,PR,Z)/*name p1,p2,...,pn.
```

LIBTASK copies all active (not logically deleted or replaced) tasks from file lfn1 onto a scratch file, copies the scratch file back onto file lfn1, and adds a new sorted task library directory to the end of file lfn1 according to input directive on the LIBTASK statement. This statement is not valid when TAF is up and has attached file lfn1.

```
LIBTASK(P=lfn1,B=lfn2,Z)/*name p1,p2,...,pn.
```

LIBTASK copies file lfn2 to the end of file lfn1 according to input directives on the LIBTASK statement. LIBTASK adds a new sorted task library directory to the end of file lfn1.

```
LIBTASK(P=lfn1,B=lfn2,PR,I=lfn3)
```

LIBTASK copies all active (not logically deleted or replaced) tasks from file lfn1 onto a scratch file, copies the scratch file back onto file lfn1, copies file lfn2 to the end of file lfn1 according to input directives on file lfn3, and adds a new sorted task library directory to the end of file lfn1. This statement is not valid when TAF is up and has attached file lfn1.

DELETING A TASK FROM A TASK LIBRARY

With the DL input directive on the LIBTASK statement, the user can logically delete a task from the task library if TAF is up and has attached the task library as follows:

```
LIBTASK(P=lfn1,TT,Z)/*name DL.
```

In this example, LIBTASK logically deletes a task (name) from file lfn1. The TT option is required when TAF is up. The DL input directive is the only input directive allowed for the named task.

With the PR option on the LIBTASK statement, the user can delete inactive tasks from the task library when TAF is not up and when TAF is up but has not attached the task library.

```
LIBTASK(P=lfn1,PR,Z)/*name DL.
```

In this example, LIBTASK physically deletes a task (name) from file lfn1. TAF must not have the file attached when this statement is executed.

IGNORING A TASK IN THE REPLACEMENT FILE

With the IG input directive on the LIBTASK statement, the user can tell LIBTASK to ignore a task binary in the replacement file specified with the B option as follows:

```
LIBTASK(P=lfn1,B=lfn2,Z)/*name IG.
```

In this example, LIBTASK skips a binary task (name) on file lfn2 when copying file lfn2 to file lfn1. The IG input directive is the only input directive allowed for the named task.

The applications programmer should be aware of the functions of these tasks but may omit the detailed information contained in this section.

Several system tasks are required with every application: initial task (ITASK), K-display directory (KDIS), logout transaction terminal (LOGT), message abort (MSABT), inactive task (OFFTASK), system message (SYSMSG), and execute named task (XTASK). Each must reside on the system task library of every transaction system. Limited versions of these system tasks are supplied by Control Data; however, modifications to these tasks are often necessary for specific applications.

The NOS Installation Handbook contains directions in the TAF installation section for installing these tasks.

INITIAL TASK (ITASK)

The primary functions of ITASK are analyzing terminal input and TAF-originated transactions. If the TAF-originated transaction field is set in the communication block (figure 2-1), ITASK uses a table to initiate the processing routine associated with that type of transaction. Otherwise, ITASK examines three characters, starting with the first character of the input message. The first of these three characters is the transaction code; the other two comprise the subtransaction code entered from a terminal.

Tables in ITASK associate combinations of transaction and subtransaction codes with task names. When a valid combination is received, the associated task name is requested for scheduling via a CALLTSK request. If an illegal combination occurs, MSABT is requested to send an error message to the terminal. It is not possible to have a transaction code without a subtransaction code entered from a terminal. However, spaces can be defined as valid subtransaction codes.

ITASK recognizes the transaction code EX., which is used to call the system task XTASK. This special transaction code is reserved for XTASK.

NOTE

Because ITASK schedules tasks, care must be taken to avoid entering an infinite loop when a task calls ITASK.

ITASK contains three tables. These tables are set up for the following codes.

- Time a task is to be run
- Transaction code
- Subtransaction code

A macro is used to set up each type of ITASK table. These three COMPASS macros (TIMCNT, TRAN, and STRAN) are discussed next.

TIMCNT MACRO

The TIMCNT macro is used to make an entry in the table of time-originating tasks (TTOT). TTOT contains the time a task is to be initiated and the address of a user-written subroutine that is called by ITASK to set up and call the desired task.

One TTOT contains all of the entries, regardless of the application that is to use them. This feature allows tasks to be run at predetermined times without specific task requests or terminal input to initiate the run.

The format of the macro is:

TIMCNT hh,mm,ss,addr

hh	Hour
mm	Minute
ss	Second
addr	Routine address

The format of the TTOT table is:

Word 1	TTOT,number of entries in the table
Word 2 and following	Table entries

The format of an entry is shown in figure 10-1.

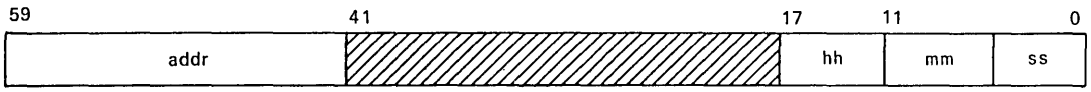
TRAN MACRO

The TRAN macro sets up entries for the table of transaction codes (TRANT). TRANT contains the transaction code and the address of the table of subtransaction codes for that particular transaction code. ITASK assumes that the TRANT table is set up in display code order. It uses the display code value of the transaction code entered by the user to locate the proper entry in TRANT. The format of the macro is:

TRAN code

code	One-character transaction code

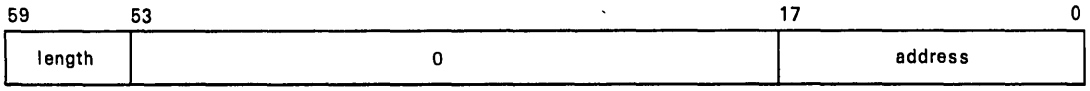
The format of an entry in TRANT is shown in figure 10-2.



addr — addr is the address of a user-written subroutine that is called by ITASK to set up and call the desired task. ITASK then clears the table entry so the request is not repeated.

hh mm ss — Time the task is to be initiated, in hours, minutes, and seconds.

Figure 10-1. Format of TTOT Table Entry



length — Length of subtransaction code table (in format allowing unpack function to retrieve it).

address — Address of section in subtransaction code table containing associated entries.

Figure 10-2. Format of TRANT Table Entry

STRAN MACRO

The STRAN macro makes entries in the table of subtransaction codes (TRANS). TRANS contains the subtransaction code and the corresponding task name. The format of the macro is:

STRAN sc, tasknam

sc — Two-character subtransaction code

tasknam — Name of the task to be called

The format of an entry in TRANS is shown in figure 10-3.

Each group of entries relating to the same transaction is placed immediately following the TRANT entry for that code, although the subtransaction table is assembled elsewhere. The following is an example of the TRANT and TRANS tables.

```
TRANT BSS     0
      TRAN    A
      STRAN   02,TSKNAME
      STRAN   03,TASKN
```

A is the transaction code, 02 and 03 are the subtransaction codes, and TSKNAME and TASKN are the respective tasks to be requested.

A site may want to delete the TRANS table entry of STRAN (X.), XTASK in ITASK for security reasons. XTASK is described in more detail later in this section.

ITASK DESCRIPTION

Entries in all ITASK tables, as well as the user subroutines for TIMCNT entries in TTOT, should be inserted in the designated places within the program before assembly. TRANT and TRANS entries must be in alphabetical or numerical order according to display codes.

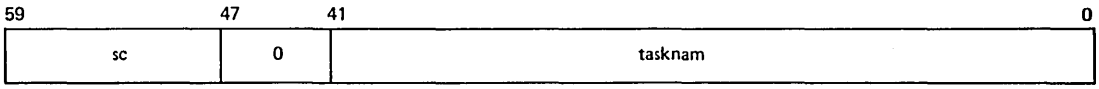
If ITASK receives a valid transaction code and subtransaction code, the call is made to the desired task. If the codes are invalid, the terminal receives an appropriate message from MSABT.

- The system time is retrieved from the appropriate word in the communication block. TTOT is then searched for any entries that were to be run by that time.
- If such an entry is found, ITASK jumps to the user's subroutine to set up and call the task. This subroutine must set up the communication block required by the task that is to be run. It must perform a CALLTSK without the cease option set to ensure that control is returned to ITASK.

TAF-ORIGINATED TRANSACTIONS

A number of conditions can cause the transaction executive to start transactions. These conditions are:

- Terminal log-in.



sc Subtransaction code.

tasknam Task name, left-justified, zero-filled.

Figure 10-3. Format of TRANS Table Entry

- K.IDLE command telling the transaction subsystem to idle down.
- Transaction executive recovery.
- Exceeding a length of time as specified by an assembly option (TACTL).

Code in common deck COMKSTC can be used to identify the condition. The condition code can be obtained by ITASK from the communication block (refer to figure 10-4). Response to these conditions depends on the application. Code furnished by the user in ITASK may process these transactions.

On executive recovery, an error field is set in the user area for a terminal or communication subsystem failure prior to ITASK receiving the TAF-origin transaction.

For a terminal log in condition, ITASK responds with the message READY and then does a normal CEASE.

Processing of a time activation of ITASK might be as follows:

1. The system time is retrieved from the appropriate word in the communication block. TTOT is then searched for any entries that were to be run by that time.
2. If such an entry is found, ITASK jumps to the user's subroutine to set up and call the task. This subroutine must set up the communication block required by the task that is to be run. It must do a CALLTSK without the cease option set to ensure that control is returned to ITASK.

KDIS

KDIS is a system task scheduled by the transaction executive when the K.SWITCH command is entered by the console operator. KDIS requests the K display and displays a complete list of K display commands. The K display commands are discussed in the NOS Operator's Guide.

LOGOUT TASK (LOGT)

This task logs a terminal out of TAF. However, the task does not log the terminal off the computer system. The task sends a message, END OF TRANSACTION SESSION, performs a LOGT request, and performs a CEASE request. The terminal is then available for login to TAF or other network applications.

MESSAGE ABORT (MSABT)

Message abort (MSABT) is a system task that sends error messages to the originating terminal when a transaction is terminated due to a system-detected error. MSABT checks for error codes that were placed in the communication block by the transaction executive or the initial task. When an error code is detected, MSABT sends the appropriate error message and initiates an abnormal CEASE function.

MSABT is scheduled in the same manner as other tasks.

If the Total data manager is used, MSABT must be modified to perform a FREEX command. This ensures that Total records are released if a task aborts.

Error messages issued by MSABT are described in appendix B.

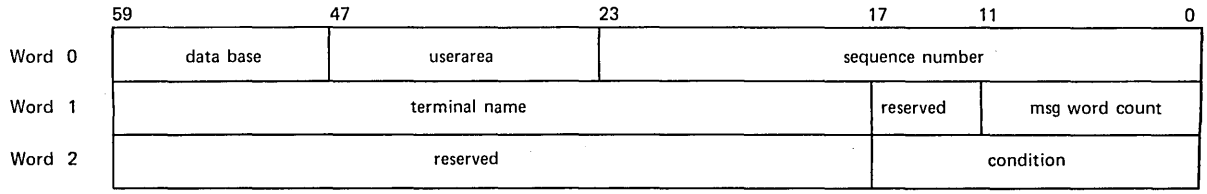


Figure 10-4. Communication Block for System Origin Transactions

OFFTASK

This task is provided to give the user control over an internal condition which might be the result of operational error. OFFTASK is scheduled by the transaction executive when a request has been received for an inactive task. The task may be inactive or off due to the use of the K.OFFTASK command or use of a LIBTASK directive when the task was placed in the library. OFFTASK sends an error message to the originating terminal and ceases abnormally. It is expected that the user will expand the task to handle any specific conditions that may exist when a transaction is interrupted by an inactive task.

SYSMSG

This task sends K.MESSAGE-entered messages and K display diagnostic messages to terminals. It calls the subroutine in COMKCBT passing the user argument bits. The word returned is sent after the message is sent.

EXECUTE NAMED TASK (XTASK)

Execute named task (XTASK) is a special task scheduler used to execute most named tasks. It will not execute tasks by the names ITASK, KDIS, MSABT, OFFTASK, SYSMSG, and XTASK. These are the names of required system tasks. XTASK can be used to schedule LOGT as a means of logging a terminal out of transaction mode. XTASK can be called via ITASK by the following input:

EX.task

EX. Special three-character transaction code which causes ITASK to schedule XTASK.

Task One- to seven-character alphanumeric task name, left-justified (no space between EX. and task), with a nonalphanumeric character terminator.

XTASK is provided as a means of scheduling tasks, and as an example of how a site might use such a task. However, it may provide too much flexibility, and therefore should be used with discretion.

The terminal interface to the transaction subsystem allows the following actions:

- Definition of a dial-up transaction terminal.
- Switching from time-sharing mode to transaction mode.
- Initiating transaction processing.
- Switching from transaction mode to time-sharing mode.
- Transmission of data to and from the transaction subsystem.

MODES OF OPERATION

The user logs into the system in the normal way. After logging in, the terminal may be switched to transaction mode (where data is being sent and received by the transaction subsystem) and back to time-sharing mode (where data is being sent and received by the time-sharing subsystem).

SWITCH TO TRANSACTION SUBSYSTEM

The time-sharing command TRAN causes input to be switched to the transaction subsystem. The format of the command is:

TRAN,terminal name

The terminal name with its preceding comma is optional. If no terminal name is provided, the default is the user number at login. The terminal name, if present, or the user number must match a dial-in entry in the network or simulation files. The user must be validated for special transaction privileges (refer to the NOS Installation Handbook) if the terminal name is different from the user number at login.

Responses to the TRAN command are:

READY

The transaction subsystem is ready to receive input from the terminal.

terminal name NOT DEFINED.

The terminal name is not in the network or simulation files.

terminal name IN USE.

The terminal name is currently being used with the transaction subsystem from another terminal.

TRANSACTION SUBSYSTEM CLOSED.

The transaction subsystem has not been activated.

INITIATE TRANSACTION PROCESSING

To initiate transaction processing, the user must enter a valid transaction and subtransaction code. The one-character transaction code must be the first character of input. The two-character subtransaction code must be the second and third characters of input. The transaction subsystem uses these transaction and subtransaction codes as described in section 10 to schedule the appropriate task. Any input following the transaction and subtransaction codes is interpreted as input to the task.

SWITCH TO TIME-SHARING SUBSYSTEM

To switch back to the time-sharing subsystem, the terminal must call a task that issues the COMPASS macro LOGT. The macro is in common deck COMKMAC and has no parameters. The system task LOGT (described in section 10) can be used to perform this function.

NOTE

The primary file is no longer available after switching back to time-sharing mode.

FORCED SWITCH TO TIME-SHARING SUBSYSTEM

If the transaction executive reinitializes and restarts after a user enters transaction subsystem mode (issues a TRAN, terminal name command from the terminal), the system switches the user back to the time-sharing mode. After the next line of terminal input is entered, the following messages are sent to the terminal: TRANEX RESTART, REENTER *TRAN* command.

This is followed by the normal time-sharing executive message for entering the subsystem the user was under before the previous TRAN command. The terminal is now in the same state as if the user had voluntarily switched from transaction mode.

The user must now enter a TRAN command to switch back into transaction mode.

NOTE

The input line entered just prior to the system malfunction message was not received by the transaction executive and must be reentered after the user switches back to transaction mode.

MESSAGE SIZE LIMITS

The maximum message size that can be received or transmitted by the transaction subsystem is determined at installation time. Default value is 640 characters total for all SENDs in a transaction. If ASCII is used, each character code requiring an escape, or shift, character counts as two 6-bit characters.

SPECIAL CHARACTERS

While in transaction mode, the I, S, or BREAK keys can be used to terminate output. The message *TERMINATE* appears, and the transaction subsystem is again ready to receive input. The I key is not an interrupt as in time-sharing mode; it operates the same as the S key.

TIME-SHARING COMMAND SUBSET

The transaction-oriented time-sharing terminal user can be limited to a subset of the time-sharing commands through the use of the subsystem TRANACT. The subsystem is specified for the user's user number in the validation file for the operating system. The system places the terminal into the TRANACT subsystem when the user logs in; there is no way to change subsystems. When under the control of the TRANACT subsystem, the only valid replies to the RECOVER/SYSTEM: message are RECOVER or TRAN.

The following commands are legal while in the TRANACT subsystem. There is no way to create a file and no way to access a file except to call a procedure file.

ASCII	LOGIN
BEGIN	NORMAL
BYE	PACKANM
CALL	PARITY
CATLIST	PASSWOR
CHARGE	RECOVER
DAYFILE	ROUT
FULL	STATUS
GOODBYE	STOP
HALF	TERM
HELLO	TIMEOUT
HELP	TRAN
LIMITS	

The user may have a requirement to handle nonstandard character sets or may want to check the parity bits associated with each character of input from the terminal. To facilitate this requirement, the user can use the binary I/O feature.

SELECTION OF BINARY I/O

A task may change the mode of the time-sharing executive by placing special control bytes in the output message. Binary input selects the mode for the next input. Binary output selects the mode for the current output. A full description of control bytes is given in an appendix to the Time-Sharing User's Reference Manual (refer to preface for publication number).

SET BINARY INPUT MODE (0006)

This byte changes the input mode from normal or ASCII to binary mode. The control byte must be byte 0. Bytes 1 and 2 are defined as follows:

Byte 1 Specifies the maximum octal number of characters to be received before input is terminated. If byte 1 is 0, the number is assumed to be 1. Byte 1 cannot exceed 150 (decimal).

Byte 2 Specifies the termination code. When a character is received from the terminal that matches the lower 8 bits of this byte, the input operation is terminated. If bit 11 of this byte is set, no termination character is assumed.

This conversion mode packs the 8 bits of data as the lower 8 bits of a 12-bit byte and sets the upper bit (bit 11). The exhaustion of the character count or the occurrence of the termination code causes the end-of-line condition to be set. A 0007 byte is forced as the first byte of input so the data is transmitted as binary if it is listed. This byte must be terminated by an end-of-line byte (0000).

INITIATE BINARY OUTPUT (0007)

This byte initiates binary output. If the user wants to output data formatted as described for binary input, a 0007 byte must precede the data. This mode continues until an end-of-line or nonbinary output data byte is detected. Control byte 0010 is appended to transaction output automatically by the time-sharing executive.

TERMINAL CONTROL

The following allows the user to maintain control over the terminal through application tasks and by modifying system tasks (refer to section 10).

- When a terminal establishes connection with the transaction executive, a special communication block is sent to ITASK. ITASK can then control the response the user gets when transaction mode is entered.
- The transaction executive does not send any messages to the terminal directly. All messages are sent either by application tasks or system tasks.
- System tasks which send messages to the terminal are:

ITASK	Sends a prompt message for connection.
LOGT	Returns terminal to time-sharing control.
OFFTASK	Sends an error message.
MSABT	Sends an error message.
SYSMSG	Sends K.MESSAGE messages and K display messages.
XTASK	Sends an error message.

All of these tasks may be modified by the user.

IDENTIFICATION OF BINARY TERMINALS

The application may select one or more of the 24 user argument (UA) bits associated with each terminal to indicate that the terminal is to be a binary terminal. This bit can be set using the UA parameter in the network file entry for the terminals or K.TST K display command. It may also be set or cleared using TARO requests from tasks.

The user argument bits of the originating terminal are passed to each task in the communication block header. They may be tested to determine if a terminal is a binary terminal within each task.

CODE PROVIDED FOR BINARY I/O

COMKCBT This is a COMPASS common deck which contains three installation parameters and a subroutine. Detailed instruction may be obtained by doing a DOCMENT of COMKCBT. The DOCMENT control statement is described in the NOS Reference Manual, volume 1 (refer to preface for publication number). The purpose of this deck is to supply a sample code for processing binary terminals. It functions as follows:

- Contains an equate for the number of the bit in the user argument area which indicates that the terminal is binary. If it is equal to 24, it indicates binary terminals are not being used. This is the default.

- Contains an equate for the character count for binary input. The default is 0.
- Contains an equate for the terminating character for binary input. The default is 4000_8.
- A subroutine is provided that uses the user argument bits as an input parameter. The subroutine returns zero if the terminal is not a binary terminal and returns a word containing the binary input control byte, character count, and terminating character if it is a binary terminal.

ITASK When the message received is a terminal connection message (0-04), ITASK passes the user argument bits from the communication block to the subroutine in COMKCBT. It then sends its READY message and does a SEND to the originating terminal of the one-word binary input request if it is a binary terminal.

MSABT After the error message has been sent, MSABT calls the subroutine in COMKCBT, passing it the user argument bits. If the word returned is nonzero, it is sent to the originating terminal as a one-word message.

SYSMSG This task sends K.MESSAGE-entered messages and K display diagnostic messages to terminals. It calls the subroutine in COMKCBT passing the user argument bits. The word returned is sent after the message is sent.

OFFTASK This task sends an error message to the originating terminal. The subroutine in COMKCBT is used to format the final SEND as in ITASK and SYSMSG.

XTASK If an error message has been sent, XTASK passes the user argument bits from the communication block to a subroutine in COMKCBT. If the word returned is nonzero, it is sent to the originating terminal as a one word message.

USER-WRITTEN CODE FOR BINARY I/O

To select binary input for each input from a terminal, the user should consider doing the following:

- Use, modify, or rewrite the subroutine in COMKCBT for the particular requirements.
- Modify ITASK to check the first byte of an input message for a 0007 byte. If present, the user must write a transaction code translation algorithm.
- An application task may either check the user argument bits or the message for a 0007 byte to determine if input is binary.
- The user may want to modify ITASK, MSABT, SYSMSG, OFFTASK, and XTASK. In modifying ITASK, it should be remembered that it is core-resident (size is important). All of the system tasks are reusable and must be self-initializing.
- The binary input request must be the first byte of an output message and thus may not be appended to a message or embedded in it.

Table A-1 lists the character codes for the 63- and 64-character sets. Users should check with a site analyst to see which character set is being used.

TABLE A-1. ASCII CODE STANDARD PRINT 63- AND 64-CHARACTER SET

ASCII CHAR.	ASCII CODE		DISPLAY CODE (OCTAL)
	OCTAL	HEXADECIMAL	
:	072	3A	00†
A	101	41	01
B	102	42	02
C	103	43	03
D	104	44	04
E	105	45	05
F	106	46	06
G	107	47	07
H	110	48	10
I	111	49	11
J	112	4A	12
K	113	4B	13
L	114	4C	14
M	115	4D	15
N	116	4E	16
O	117	4F	17
P	120	50	20
Q	121	51	21
R	122	52	22
S	123	53	23
T	124	54	24
U	125	55	25
V	126	56	26
W	127	57	27
X	130	58	30
Y	131	59	31
Z	132	5A	32
0	060	30	33
1	061	31	34
2	062	32	35
3	063	33	36
4	064	34	37
5	065	35	40
6	066	36	41
7	067	37	42
8	070	38	43
9	071	39	44
+	053	2B	45
-	055	2D	46
*	052	2A	47
/	057	2F	50
(	050	28	51
)	051	29	52
$	044	24	53
=	075	3D	54

† IN THE 63-CHARACTER SET, THIS DISPLAY CODE REPRESENTS A NULL CHARACTER. ALSO, USE OF THE COLON IN PROGRAM AND DATA FILES MAY CAUSE PROBLEMS. THIS IS PARTICULARLY TRUE WHEN IT IS USED IN PRINT AND FORMAT STATEMENTS.

3AE15A

TABLE A-1. ASCII CODE STANDARD PRINT 63- AND 64-CHARACTER SET (Contd)

ASCII CHAR	ASCII CODE		DISPLAY CODE (OCTAL)
	OCTAL	HEXADECIMAL	
(SPACE)	040	20	55
,	054	2C	56
.	056	2E	57
#	043	23	60
[	133	5B	61
]	135	5D	62
%	045	25	63†
"	042	22	64
—	137††	5F	65
!	041	21	66
&	046	26	67
'	047	27	70
?	077	3F	71
<	074	3C	72
>	076	3E	73
@	100	40	74
\	134	5C	75
^	136	5E	76
;	073	3B	77

† IN THE 63-CHARACTER SET, THIS DISPLAY CODE REPRESENTS A COLON (:), 7-BIT ASCII CODE 072, 7-BIT HEXADECIMAL CODE 3A.

†† ON TTY MODELS HAVING NO UNDERLINE, THE BACKARROW (←) TAKE ITS PLACE.

3AE16A

ERROR MESSAGES B

This appendix contains an alphabetical listing of messages which may be important to the TAF programmer. Each message is followed by an explanation of the message and/or the circumstances causing it to be issued, the recommended action, and the routine which issued the message.

Lowercase letters are used within a message to identify variable fields. All messages beginning with lowercase (variable) fields are listed alphabetically according to the first nonvariable field after the messages that begin with nonvariable fields.

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
AAM FILES CLOSED.	TAF CRM has closed all AAM files.	None.	TAF
ABORT OF CDCS DETECTED.	Self-explanatory.	None.	TAF
APPLICATION ERROR - NO TERMINAL OUTPUT.	A task error. A transaction has run without either an error message or a SEND being sent to the originating terminal.	Correct task or inform site analyst.	MSABT
ARITHMETIC OR OTHER SOFTWARE ERROR IN TASK.	A task error. Hardware or the system detected a software error in the task.	Correct task or inform site analyst.	MSABT
ATTACH ERROR ON - filenam.	The transaction executive cannot attach the file filenam under present conditions. This usually implies that the file does not exist or permission has not been given to the TAF user name.	Correct error and reinitialize executive, or contact site analyst.	TAF
ATTACH MODE MUST BE W, M, R, OR RM.	The mode parameter on the CRM statement must be one of the specified values.	Correct the mode parameter on the CRM statement or inform site analyst.	TAF
BINARY RECORD WAS LESS THAN 64D WORDS LONG.	The task binary file contains records in the wrong format.	Ensure that a LOAD was performed after compilation of the tasks.	LIBTASK
BINARY RECORD WAS NEITHER ABSOLUTE NOR OVERLAY FORMAT.	The task binary file contains records in the wrong format.	Ensure that a LOAD was performed after compilation of the tasks.	LIBTASK
BLOCKAGE AMONG CM RESIDENT TASKS.	The sum of initial field lengths for the CM resident tasks exceeds the minimum size of total task area.	Correct error.	TAF
CALLRTN NESTED CALL LIMIT EXCEEDED.	A task error. An attempt was made to nest CALLRTN requests more than 16 levels.	Correct task or inform site analyst.	MSABT
CDCS NOT AVAILABLE.	Your transaction attempted to use CDCS when it was not present in the system, CDCS aborted while your transaction was waiting for completion of a CDCS request (SSC), or CDCS aborted while your transaction was rolled out.	Inform data base administrator.	MSABT
CDCS REQUESTED ABORT.	TAF aborted the transaction as a result of a request by CDCS. The detailed error message returned by CDCS is available in the JOUR0 file.	Inform data base administrator.	MSABT
CHANGED TLD DETECTED - filenam, usernam.	An unrecognizable library directory format was encountered during a library directory update attempt.	Inform site analyst.	TAF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
CONFLICTING LIBTASK OPTIONS - p1,p2.	The user specified an illegal combination of LIBTASK control statement options. Possible illegal combinations of p1 and p2 are: CR,PR CR,TT PR,TT TT,N Another possible illegal combination gives P=0,-N for p1,p2. This means that the user specified P=0 but did not specify N.	The user must choose either p1 or p2. For the P=0,-N combination, the user must specify N if no old task library (P=0) exists.	LIBTASK
CONTROL CARD ARGUMENT ERRORS.	Errors exist in the parameters on the LIBTASK control statement.	Correct and rerun.	LIBTASK
CRM(...parameter-list...)	This is a copy of a CRM statement that is in error. A subsequent message follows.	Inform site analyst.	TAF
CRM DATA MANAGER SUCCESSFULLY LOADED.	Self-explanatory.	None.	TAF
CSM - ILLEGAL COMMUNICATION FUNCTION.	An illegal or unrecognizable request was received by the transaction executive from the CPU monitor.	Inform site analyst.	TAF
DATA MANAGER NOT LOADED.	The TAF data manager was called but has not been loaded.	Inform site analyst.	MSABT
DIRECTIVE ERRORS.	Dayfile message indicating that one or more input directives were in error. Fatal error.	Examine output file to determine reason for error.	MODIFY OPLEDIT LIBTASK MODVAL PROFILE SYSEDIT
DIRECTORY IS FULL-NO MORE ADDITIONS ALLOWED.	The maximum length of the task library directory has been reached.	LIBTASK must be reassembled to allow for a larger directory.	LIBTASK
DISPLAY DUMP NOT ALLOWED TO TERMINAL.	A display code dump to a terminal is not allowed.	Specify the O option on the KTSDMP control statement if a dump to a terminal is desired.	KTSDMP
DSP ERROR ec RETURNED ON JOB ROUTE.	DSP has informed the transaction subsystem that an error was detected in routing a job for task dump processing. ec is the DSP error code.	Inform site analyst.	TAF
DSP ERROR ec RETURNED ON TASK SUBMIT.	DSP has informed the transaction subsystem that an error other than a job or user statement error was detected. ec is the DSP error code.	Inform site analyst.	TAF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
DUAL AND TRACE FLAGS FOR FILE filenam.	It is illegal to dual-record and trace the same file.	Correct error and reinitialize executive (TAF) or rerun job (BDMI).	TAF BDMI
DUPLICATE DATA BASE IN TCF - xx.	Active data base identifier, xx, in the TCF is not unique.	Fix TCF so that xx appears only once among active (ON) DMS statements.	TAF
ECS READ ERROR.	Self-explanatory.	Inform customer engineer.	TAF
ECS TASK tasknam NOW MS RESIDENT.	Task tasknam could not be loaded into ECS because of insufficient storage. It is loaded into mass storage.	If task must be resident in ECS, more ECS space must be allocated for the TAF user name. Refer to the NOS Installation Handbook.	TAF
ECS WRITE PARITY ERROR ENCOUNTERED.	Self-explanatory.	Inform customer engineer.	TAF
EDT CARD NOT USED FOR CRM.	The xxJ file cannot contain an EDT statement for TAF CRM.	Remove the EDT statement from the xxJ file or inform site analyst.	TAF
ERROR IN LOADING AAMI.	The loader encountered errors while loading the TAF CRM AAM interface (AAMI).	The site analyst should consult the CYBER Loader Reference Manual (listed in the preface).	TAF
ERROR IN LOADING HASH CODE filenam.	The loader encountered errors while loading the hashing routine code that is on file filenam.	The site analyst should consult the CYBER Loader Reference Manual (listed in the preface).	TAF
ERROR IN LOADING TOTAL.	The loader encountered errors while loading Total and the data base descriptor modules (DBMODs).	The site analyst should consult the CYBER Loader Reference Manual (listed in the preface).	TAF
ERROR IN READING TASKLIB-filenam.	Error occurred during transaction executive initialization or ECS-resident task loading. File specified as task library was incorrectly formatted; therefore, it could not be read or loaded into ECS correctly.	Inform site analyst.	TAF
ERROR IN SUBMITTED FILE.	An illegal parameter has been detected on a SUBMT call from a task.	Correct task or inform site analyst.	MSABT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ERROR ON xxJ FILE ARGUMENTS.	The xxJ file contains statements in error, which causes the transaction subsystem to abort.	Examine xxJ file. Consult TAF data base administrator.	TAF
FATAL CIO ERROR STATUS.	A TAF CIO operation returned a fatal error status which aborted TAF.	Inform site analyst.	TAF
FIELD LENGTH EXCEEDED FOR CMM.	TAF does not have enough field length to allocate the space potentially required by CMM.	Increase TAF initialization field length.	TAF
FIELD LENGTH EXCEEDED FOR LOCKS.	TAF does not have enough initialization field length for allocating lock tables.	Decrease the locks parameter on the CRM statement, increase the TAF initialization field length, or inform site analyst.	TAF
FIELD LENGTH EXCEEDED FOR RECORD.	TAF does not have enough field length to allocate the space for the record buffer.	Decrease the record size specified in the xxJ file or increase the TAF initialization field length.	TAF
FIELD LENGTH EXCEEDED FOR USERS.	TAF does not have enough initialization field length for allocating file control tables.	Decrease the users parameter on the CRM statement, increase the TAF initialization field length, or inform site analyst.	TAF
FILE edt EMPTY.	Element descriptor table file edt is empty.	Correct error, reinitialize executive, and rerun.	TAF BDMI
FILE xxJ NOT FOUND.	Transaction subsystem aborts. Data base in TCF file has no xxJ file, or a PFM error occurred.	Consult TAF data base administrator or contact site analyst.	TAF
FILE NAME CONFLICT.	The input file name specified on the KTSDMP control statement is the same as the output file name specified.	Correct error and rerun.	KTSDMP
FILE NAME MUST BE 2-7 CHARACTERS.	The xxpfni parameter on the CRM statement must be two to seven characters, the first two (xx) being the data base name.	Correct the xxpfni parameter on the CRM statement or inform site analyst.	TAF
FILE NOT ATTACHED - filenam.	Either the P or the N parameter file was not attached when the DA option was used; filenam is the permanent file name.	Ensure that the file is defined or attached prior to entering the LIBTASK statement.	LIBTASK

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
FILE hash NOT FOUND.	The indirect file named hash containing the binary code of the hashing routine was not found under the usernam parameter on the USER statement in the xxJ file or a PFM error occurred.	Ensure that file hash is saved under the usernam parameter or inform site analyst. Consult the CYBER Loader Reference Manual (listed in the preface).	TAF
FILE TCF EMPTY.	An empty TCF exists under the TAF user name.	Place the necessary information on TCF.	TAF
FILE TCF NOT FOUND.	The TCF was not found under the user name of the Transaction Facility.	Create a TCF file under the TAF user name.	TAF
FILE edt TOO LARGE.	Element descriptor table is too large to be read into memory.	Correct error and reinitialize executive (TAF), or rerun (BDMI).	TAF BDMI
FILE TYPE MUST BE IS OR DA.	The type parameter on the CRM statement must be either IS (indexed sequential) or DA (direct access).	Correct the type parameter on the CRM statement or inform site analyst.	TAF
FL REQUEST BEYOND MFL (CM).	CM field length requirements for the task exceed the CM field length allowed.	Contact data base administrator.	MSABT
FL TOO LARGE- nnnnnnB,tasknam,tasklib.	The initial load field length, nnnnnnB, for task tasknam on task library tasklib exceeds the minimum size of the transient task area (potential space available to contain transient tasks). Thus a situation could arise in which it would not be possible to load the task.	Correct error.	TAF
FORMAT ERROR IN TERMINAL DESCRIPTION FILE.	Statements on the NETWid file are in error.	Run VALNET on NETWid. Correct indicated errors. Reinitialize transaction subsystem.	TAF
FORMAT ERROR IN THE NETWORK DESCRIPTION FILE.	During transaction executive initialization, one or more errors were found to exist in the network description file.	Inform site analyst.	TAF
FWA .GE. LWA+1.	There is a logical error in the structure of the input file which implies that the first word address is greater than or equal to the last word address plus one.	Inform site analyst.	KTSDMP
ILLEGAL COMMON MEMORY MANAGER REQUEST.	Memory request with reserved bits in the parameter block set incorrectly.	Do not set reserved bits.	MSABT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ILLEGAL DATA BASE IN xxJ FILE.	One of the statements in the xxJ file specifies an incorrect xx parameter and causes the transaction subsystem to abort.	Examine xxJ files. Consult the TAF data base administrator.	TAF
ILLEGAL FWA FOR LOAD.	The FWA of the load was not equal to 111B. Task binaries must be in absolute format.	Ensure that the correct load procedure is performed.	LIBTASK
ILLEGAL JOURNAL FILE REQUEST.	Either a task requested that an entry be made on an undefined journal file, or the task attempted to make an entry longer than 2499 CM words on a journal file.	Inform site analyst.	MSABT
ILLEGAL LIBTASK ATTEMPT - filenam, usernam.	The transaction executive validates all dynamic attempts to change the task library by comparing the user number of the list of the requester against data base user numbers. If it does not match, or if the library file is not attached by TAF, the transaction executive issues this dayfile message, where usernum is the user number of the illegal attempt.	Correct and reinitialize transaction executive.	TAF
ILLEGAL NETWORK CALL.	A task issued a BLDABH, GETABH, SETCHT, or TERMDEF command, which is invalid under TAFTS.	Correct task or inform site analyst.	MSABT
ILLEGAL NUMBER FOR LOCKS.	The locks parameter on the CRM statement is in error. One of the following format conditions exists. - A nonnumeric character. - A character after a postradix of B or D. - An 8 or 9 with a postradix of B.	Correct the locks parameter on the CRM statement or inform site analyst.	TAF
ILLEGAL NUMBER FOR USERS.	The users parameter on the CRM statement is in error. One of the following format conditions exists. - A nonnumeric character. - A character after a postradix of B or D. - An 8 or 9 with a postradix of B.	Correct the users parameter on the CRM statement or inform site analyst.	TAF
ILLEGAL PARAMETER IN RA+1 CALL.	A task error. The parameters specified in a request from a task are either insufficient or invalid.	Correct task or inform site analyst.	MSABT
ILLEGAL REDUCTION OF FL.	Task request to reduce FL would cause the last word address of the communication block to be outside the task FL.	Contact data base administrator.	MSABT
ILLEGAL SUBTRANSACTION CODE - INITIAL TASK.	There is an unrecognizable subtransaction code in the terminal input.	Correct terminal input or inform site analyst.	MSABT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
ILLEGAL TASK NAME REQUESTED FOR SCHEDULING.	The task requested was not found in the task library directory.	Correct task or inform site analyst.	MSABT
ILLEGAL TERMINAL INPUT REQUEST.	A task error. A WAITINP request was issued by a memory resident task or by a task that did not originate from a terminal.	Correct task or inform site analyst.	MSABT
ILLEGAL TERMINAL NAME.	A batch job submitted a transaction specifying a nonexistent terminal and/or user name.	Correct task or correct and reinitialize transaction executive with terminal and user name defined.	TAF
ILLEGAL TOTAL INTERLOCK REQUEST.	One of the following. - A task that is not on the system task library issued a CHKON request. - A task issued a CHKON request with a CHKON request already in effect. - A task issued a CHKOFF request without first issuing a CHKON request. - A task that issued a CHKOFF request had a sequence number different from a task that issued a CHKON request.	Inform site analyst.	MSABT
ILLEGAL WORD COUNT ON TERMINAL OUTPUT.	One or more of the following task errors occurred on a SEND request. - The word count was less than or equal to zero. - There was an attempt to use SEND MAXWS or more CM words of information in one SEND; MAXWS is an installation option. - There was an attempt to use SEND MAXTO or more CM words of information by one task through a series of SEND requests. MAXTO is an installation option.	Correct task or inform site analyst.	MSABT
INACTIVE TASK REQUESTED.	An attempt was made to schedule a task which has been turned off with a K.OFFTASK command or a LIBTASK directive.	Correct condition or inform site analyst.	OFFTASK
INITIAL TASK NOT IN TASK LIBRARY DIRECTORY.	The task library file does not contain the initial task (ITASK).	Inform site analyst.	TAF
INITIALIZATION OPTIONS.	This message precedes messages indicating the values of the initial K-display options either during initialization or recovery.	None.	TAF
INSUFFICIENT FL FOR DATA MANAGER.	The transaction executive requires more field length at initialization time than is available.	Correct error and reinitialize executive.	TAF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
INVALID DATA BASE NAME ON DMS STATEMENT.	A data base name associated with TAF, CRM, or OTHER exceeds two characters.	Correct the DMS statement on TCF file.	TAF
INVALID TASK NAME.	The error message was issued for one of the following reasons: - Initialization of a task by the name of ITASK, KDIS, MSABT, OFFTASK, SYSMSG, or XTASK was attempted using XTASK. These are names of required system tasks and will not be initiated by XTASK. - A zero length task name was specified. - A task name of more than seven characters was specified.	Correct the task name.	XTASK
INVALID TCF ENTRY.	A syntax error was detected on the TCF file. This could include one of the following: - The data manager identifier is not TAF, CRM, TOTAL or OTHER. - A data manager status declaration is not in the proper form; it must be ON or OFF.	Correct error.	TAF
INVALID TOTAL PARAMETER LIST.	Parameter of Total is outside the field length.	Inform site analyst.	MSABT
JOURNAL TYPE DOES NOT MATCH xxJ FILE.	Journal file entries in the xxJ file do not match the files themselves. This causes the transaction subsystem to abort.	Consult TAF data base administrator. Examine xxJ file for the journal file entries.	TAF
K.CMB=nn. K.ECS=nnnK. K.MDM=n. K.MFL=nnnnnnB. K.REC=a. K.SCP=nn. K.TLF=a.	Values of the initial K-display options at either initialization or recovery.	None.	TAF
K-DISPLAY COMMAND UNSUCCESSFUL, FORMAT ERROR--command.	There is an error in the syntax of the command or in one of the parameters. The command field is the attempted command or blanks if the command is longer than seven characters or contains an illegal character.	Correct the command and retry the operation.	SYSMSG
K-DISPLAY COMMAND UNSUCCESSFUL, ILLEGAL COMMAND--command.	The command is not a valid K-display command or was misspelled. The command field is the attempted command.	Correct the command and retry the operation.	SYSMSG
K-DISPLAY COMMAND UNSUCCESSFUL, SYSTEM BUSY--command.	The system is busy and cannot respond to the request. The command field is the attempted command.	Retry the operation.	SYSMSG

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
K-DISPLAY COMMAND UNSUCCESSFUL, DROP IGNORED--command.	A K.DROP or K.DDROP could not be processed because the task was in recall, the time-sharing executive denied a request, or a command was issued during the initial load of the task. The command field is the attempted command or request; blanks indicate no command was issued.	Reenter the K.DROP or K.DDROP command. When the recall operation, time-sharing request, or initial load operation is completed, TAF accepts the command and aborts the task.	SYSMSG
K-DISPLAY COMMAND UNSUCCESSFUL, TELEX BUSY--command.	The time-sharing executive is busy and cannot respond to the request. The command field is the attempted command.	Retry the operation.	SYSMSG
K.MAXFL,nnnnnnB.	The run-time K-display command K.MAXFL was entered with the indicated value.	None.	TAF
K.MAXFL REJECTED.	A value was entered which caused potential blocked tasks to be detected.	Reenter K.MAXFL with a larger value.	TAF
KTSDMP COMPLETE.	Dump completed normally.	None.	KTSDMP
LIBRARY DIRECTORY EMPTY - filenam.	The task library file indicated does not contain a directory.	Inform site analyst.	TAF
LIBRARY DIRECTORY ERROR - filenam.	The task library file indicated contains a nonrecognizable directory.	Inform site analyst.	TAF
LIBRARY DIRECTORY TOO LONG - filenam.	The directory record on the task library file indicated exceeded the maximum length allowed by the transaction executive (398 entries).	Inform site analyst.	TAF
LIBRARY FILE TIME OUT.	The library file was busy for the length of time specified by an assembly constant, so LIBTASK was unable to attach it. Another user apparently has the file attached and will not allow modification.	Retry at a later time.	LIBTASK
LIBTASK-PFM ERROR DETECTED.	A permanent file error other than FILE BUSY was received when the library file attach was attempted.	Correct and rerun.	LIBTASK
LOADING ECS tasknam.	Informative message. The transaction subsystem is loading task tasknam.	None.	TAF
MEMORY OVERFLOW DURING INITIALIZATION.	TAF aborted because its field length for initialization was insufficient.	Inform site analyst. IFL= in deck TAF should be increased. Increasing the central memory field length parameter on the RFL control statement in the TAF initialization procedure file (ffff) does not correct this	TAF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		problem.	
MEMORY REQUEST FOR ECS NOT ALLOWED.	Illegal request of ECS by task.	Contact data base administrator.	MSABT
MEMORY REQUEST NOT ALLOWED WITH DATA MANAGER REQUESTS OUTSTANDING.	A task error. MEM or RFL request attempted with data manager requests outstanding.	Correct task or inform site analyst.	MSABT
MFL TOO LARGE - nnnnnnB,tasknam,tasklib.	The MFL (initial field length plus expandable field length) of the non-CM resident task (tasknam) on task library (tasklib) exceeds the minimum size of the transient task area (potential space available to contain transient tasks). Thus a situation could arise in which it would not be possible to complete processing of this task.	Reduce the task FL or EF, or increase the TAF FL.	TAF
MINIMUM TAF MFL NEEDED = nnnnnnB.	Potentially blocked tasks were detected at one of the following times: - TAF initialization - Attempted task library update - Attempt to change TAF maximum FL via K.MAXFL command. The above operation did not complete normally. The maximum FL of TAF must be at least nnnnnnB. If nnnnnnB exceeds the largest field length possible for TAF (377700B), then other corrective action is needed.	Correct error.	TAF
MISSING HEADER WORD ON xxJ FILE.	The first statement on the xxJ file is in error, causing the transaction subsystem to abort.	Examine xxJ files for header xxJ. Consult the TAF data base administrator.	TAF
MORE THAN FIVE TASKS IN TASK CHAIN.	A task error. Only five tasks can be requested for scheduling in one CALLRTN, CALLTSK, or NEWTRAN request; more than five were requested.	Correct task or inform site analyst.	MSABT
MORE THAN ONE ENTRY POINT ON A *54* TABLE.	Multiple entry points are not allowed in tasks.	Rewrite the task with only one entry point.	LIBTASK
NEED AT LEAST xx SUBCONTROL POINTS.	There are more CM resident tasks defined than subcontrol points. If non-CM resident tasks exist, there must be at least one more subcontrol point than there are CM resident tasks.	Reinitialize the transaction executive and assign more subcontrol points, or reduce the number of CM resident tasks.	TAF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
NETWORK DESCRIPTION FILE NOT FOUND.	A NETWid file is not present under the system user index for terminal communications.	Inform site analyst.	TAF
NETWORK SHUT DOWN DETECTED.	Self-explanatory.	None.	TAF
NO ACCOUNT/USER CARD IN xxJ FILE.	The ACCOUNT or USER statement in the xxJ file is not present, causing the transaction subsystem to abort.	Add ACCOUNT or USER statement in xxJ file. Consult the TAF data base administrator.	TAF
NO CRAS TERMINAL DEFINED.	Informative message indicating that no CRAS terminal is defined in the terminal validation file.	None.	TAF
NO DATA BASE ID FOR DATA MANAGER.	At least one data base identifier must be specified on each active (ON) DMS statement.	Add data base identifier to DMS statement(s) or specify status as OFF.	TAF
NO DATA BASE NAME IN xxJ FOR TOTAL.	Self-explanatory.	Add data base name to xxJ file.	TAF
NULL DESCRIPTION FILE.	Self-explanatory.	Create a description file (NETWid, SIMFid, or NCTFid where id is the machine identifier).	TAF
OFF TASK tasknam-LIBRARY libnam.	Task tasknam in task library libnam could not be loaded from ECS or recovered and loaded from mass storage. Task was turned off. Transactions using tasks will abort.	Inform site analyst. Library must be recreated.	TAF
OPERATOR IDLE DOWN.	The operator dropped TAF.	None.	TAF
OPERATOR STOP.	Informative message indicating that the operator has stopped TAF.	None.	TAF
POOL FILE - filenam NOT FOUND IN EDT.	The pool file xxERPF contains a record for file filenam not defined in the EDT for data base xx. A CRAT entry is not made for the record for this file.	Inform site analyst.	BDMI TAF
POSSIBLE BLOCKAGE AMONG CM RESIDENT TASKS.	The sum of the maximum field lengths (MFLs) for the CM resident tasks exceeds the minimum size of the total task area (potential space available to contain tasks). Thus one or more CM resident tasks could be blocked from completing.	Correct error.	TAF
POTENTIALLY BLOCKED TASKS DETECTED.	During TAF initialization, potentially blocked tasks were detected. Preceding error messages contain additional details.	Correct error.	TAF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
RA+1 CALL ERROR.	There was an unrecognizable task request.	Correct task or inform site analyst.	MSABT
RA+1 CALL PARAMETER ADDRESS OUTSIDE FL.	A parameter in a task request was outside the field length of the task.	Correct task or inform site analyst.	MSABT
RA+1 CALL WITH ILLEGAL FUNCTION CODE.	There was an unrecognizable task request for system action.	Correct task or inform site analyst.	MSABT
RECOVERY COMPLETE.	The transaction executive or time-sharing subsystem has successfully completed recovery.	None.	TAFNAM2 TAFTS2 IAFEX TELEX
RECOVERY IMPOSSIBLE.	A fatal error occurred during transaction executive initialization or the recovery file prepared by the TAF initialization routine does not exist or could not be read.	Check dayfile for a specific error message or inform the site analyst.	TAFNAM2 TAFTS2
RECOVERY IN PROGRESS.	The time-sharing subsystem or the transaction executive has begun recovery procedures due to an abort or termination condition.	None.	TAFNAM2 TAFTS2 IAFEX TELEX
REFERENCE MADE TO ILLEGAL TERMINAL NAME.	The terminal name or user name specified has not been defined.	Correct task or inform site analyst.	MSABT
REQUEST NOT ALLOWED WITH DATA MANAGER REQUESTS OUTSTANDING.	A task error. CALLRTN, WAIT or WAITINP request was attempted with data manager request(s) outstanding.	Correct task or inform site analyst.	MSABT
REQUESTED ECS NOT AVAILABLE.	The amount of ECS requested was not available in a contiguous block.	Reinitialize with less ECS requested.	TAF
SEND TO TERMINAL NOT LOGGED IN.	A SEND request was issued to a terminal not currently logged into TAF.	Correct error condition or inform site analyst.	MSABT
SYSTEM ABORT OF TASK.	The operator aborted the task (via a K.DROP or K.DDROP command), or a data manager error caused the task to abort.	Inform site analyst.	MSABT
TABLE OVERFLOW-TOO MANY TASKS.	The managed tables used by LIBTASK are not large enough for the number of tasks or directives.	LIBTASK must be reassembled with larger table allocation.	LIBTASK
TAF DATA MANAGER SUCCESSFULLY LOADED.	Self-explanatory.	None.	TAF
TAF INTERNAL ERROR.	TAF has found internal data to be inconsistent.	Perform a dump of TAF or inform site analyst.	TAF
TAF NOT CALLED BY DSD.	The transaction subsystem aborts if initiated in any way other than DSD command.	None.	TAF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
TAFNAM/TAFTS TERMINATE.	Transaction subsystem has terminated.	None.	TAF
TASK EDITING COMPLETE.	Dayfile message indicating that the run has terminated. Normal LIBTASK output consists of four lists - the input directives, the tasks on the current library, the tasks added to the library, and the tasks replaced on the library. After these lists, the time and date of the last library modification are given. If errors occurred during processing of the task binary file, a list of error messages also appears. If the LIBTASK run was initiated from a terminal, the directive and current library lists do not appear.	Refer to next dayfile message.	LIBTASK
TASK LIBRARY DIRECTORY EMPTY.	The last record of the library file is empty.	Ensure that the correct file has been specified; if so, the library must be recreated.	LIBTASK
TASK LIBRARY DIRECTORY EMPTY - libnam.	The file specified as the task library contains no recognizable directory. TAF aborts.	Inform site analyst. Task library libnam must be corrected and TAF reinitialized.	TAF
TASK LIBRARY DIRECTORY ERROR - libnam.	Task library libnam contains no recognizable directory. TAF aborts.	Inform site analyst. Task library must be corrected.	TAF
TASK LIBRARY DIRECTORY ERRORS.	The first word of the last record on the library file does not contain the directory header.	Correct and rerun.	LIBTASK
TASK LIBRARY DIRECTORY TOO LONG.	The directory is too long for the buffer allocated.	This may be altered by changing an assembly constant.	LIBTASK
TASK LIBRARY DIRECTORY TOO LONG - libnam.	The directory record on task library libnam exceeded the maximum length allowed by the transaction executive (170 files). TAF aborts.	Inform site analyst. Size of task library libnam must be reduced and TAF reassembled.	TAF
TASK NOT VALIDATED FOR REQUEST.	One of the following actions has occurred. - The terminal operator initiated a transaction which tried to perform an action associated with a data base for which the terminal was not validated. - A NEWTRAN request was issued by a task not in the system task library (TASKLIB).	Perform the appropriate action. - Inform site analyst; transaction must be reinitialized. Set up the terminal name in the network file to use the data base. The system data base (SY) may be used.	MSABT

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
		- Put the task on TASKLIB.	
TASK REQUEST ARGUMENT ERROR.	A parameter error occurred in a request.	Correct task or inform site analyst.	MSABT
TASK REQUESTED CEASE BEFORE DM FINISHED.	The task requested to end before the data manager completed all outstanding requests.	Correct task or inform site analyst.	MSABT
TASK TIME LIMIT.	The task exceeded n time slices. The length of the time slice is set by an installation parameter. The number of time slices, n, allowed for the execution of a task can be changed with a task ITL request.	Increase time slices or inform site analyst.	MSABT
TERMINAL termid Ccc Ee Pp MALFUNCTION.	Terminal failed to answer poll. termid Terminal identifier cc Multiplexer channel number e Equipment number p Port number	Inform customer engineer. This message may be indicative of hardware failure.	TAF
TERMINAL termid Ccc Ee Pp UP.	Terminal is operative after previously failing to answer. termid Terminal identifier cc Multiplexer channel number e Equipment number p Port number	Inform customer engineer. This message may be indicative of hardware failure.	TAF
TERMINALS MISSING IN NETWORK FILE.	A valid network file was found but no transaction terminals were defined in it. If TAF interfacing with TELEX is being used, TELEX was not reinitialized to recognize the same network file that TAF was using.	Ensure that the network file is under the proper name; if it is, inform site analyst. If TAF interfacing with NAM is being used, the network file is NCTFid. If TAF interfacing with TELEX is being used, the network file is NETWid if TAF/TS can attach it; otherwise, TAF/TS uses SIMFid. For the TELEX interface, reinitialize TELEX and TAF. For the NAM interface, reinitialize TAF.	TAF
TLD OVERFLOW.	Space is reserved at initialization to dynamically add tasks to the TAF resident copy of each task directory. The number of tasks is limited (default is 10) by the installation parameter TLDL. This message indicates that more tasks than allowed have been added since the most recent TAF initialization. The extra tasks will be	None.	TAF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	ignored by TAF until it is reinitialized.		
TOO MANY BRANCHES IN TASK CHAIN REQUESTED.	Only three branches are allowed in the task chain; more than three were requested.	Correct task or inform site analyst.	MSABT
TOO MANY CRAT FILE ENTRIES.	There are too many error records on the xxERPF file.	Inform site analyst. The bad areas on disk should be flawed and the data base reloaded.	TAF
TOO MANY DATA BASE NAMES.	The number of data base names associated with one data manager via DMS statements exceeds the value of MAXDB.	Decrease the number of data base names associated with the data manager.	TAF
TOO MANY FILES IN TOTAL DATA BASE.	Self-explanatory.	Reduce the number of entries in the TCF file or increase TMAXFIL.	TAF
TOO MANY JOURNAL FILES IN xxJ FILE.	More than three journal files per data base were specified, causing the transaction subsystem to abort.	Examine xxJ file for xxJOR entries. Consult the TAF data base administrator.	TAF
TOO MANY RA+1 CALLS.	Only n RA+1 requests are allowed; more than n occurred, where n is an assembly constant.	Inform site analyst. Possibly a task is in a loop.	MSABT
TOTAL DATA MANAGER NOT LOADED.	Total data manager was called but has not been loaded.	Inform site analyst.	MSABT
TOTAL DATA MANAGER SUCCESSFULLY LOADED.	Self-explanatory.	None.	TAF
TOTAL DID NOT RECOVER PROPERLY. STATUS IS yyyy.	An error status yyyy was returned on a Total FINAL call. Refer to Diagnostics in the Total Reference Manual for yyyy.	Correct error and reinitialize transaction executive.	TAF TAFNAM2 TAFTS2
TRANSACTION RESTART, REENTER *TRAN* COMMAND.	The time-sharing and transaction subsystem have completed recovery procedures for the terminal.	To reenter the transaction subsystem enter the TRAN command.	TAF
TRN - ABNORMAL.	A TAF-originated task could not be scheduled because of a lack of communication blocks or ITASK queue was full. An attempt to schedule the task will be made at a later time.	Inform site analyst.	TAF
TT OPTION REQUIRES USER NUMBER.	When updating a task library on-line (TT option is specified on LIBTASK statement), the user number must be specified prior to the LIBTASK statement so the library associated with that user number can be found.	Specify user number via USER or ACCOUNT statement before LIBTASK statement and rerun job.	LIBTASK

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
UN=usernum NOT VALID ON FM=family.	User number usernum on the specified family is not valid. The user number and family may not be defined or are incorrect in xxJ file.	Inform site analyst.	TAF
UNABLE TO ATTACH TOTAL BINARIES.	File of Total binaries is not under the user index of the transaction subsystem or a PFM error occurred.	Correct error and reinitialize transaction executive or contact site analyst.	TAF
UNABLE TO ATTACH TOTAL DBMOD BINARIES.	One or more of the DBMOD files listed on the TOTAL DMS control statement in the TCF file could not be attached under the user index of the transaction subsystem or a PFM error occurred.	Correct error and reinitialize transaction executive or contact site analyst.	TAF
UNKNOWN FILE FORMAT.	There is a logical error in the structure of the input file. It does not conform to the established format rules.	None.	KTSDMP
UNKNOWN FORMAT ON ENTRY POINT TABLE.	The task binary file contains records in the wrong format.	None.	LIBTASK
UNRECOGNIZABLE CIO ERROR CODE.	Self-explanatory.	Inform site analyst.	TAF
WAITINP FROM MULTI QUEUED TASK.	A WAITINP request was issued by a task that was scheduled to process more than one transaction or by a task that was called with return.	Correct task or inform site analyst.	MSABT
n.nnn AVERAGE ACTIVE SUBCONTROL POINTS.	Average number of simultaneously active subcontrol points when TAF is not rolled out. An active subcontrol point is one which is in recall, is waiting to use the CPU, or is currently assigned the CPU. The sampling rate is once per second.	None.	TAF
n.nnn AVERAGE OUTSTANDING CDCS REQUESTS.	Average number of outstanding (uncompleted) SSC requests per second. The sampling rate is once per second.	None.	TAF
xxxxxxx IN USE.	Another user is logged in using terminal xxxxxxx.	Retry later.	TELEX
n.nnn KILO CDCS REQUEST REJECTS FOR BUSY.	Total number of SSC rejects for busy when less than seven outstanding CDCS SSC requests existed at the time of the current request.	None.	TAF
n.nnn KILO CDCS REQUEST REJECTS FOR MAXR.	Total number of SSC attempts when there were seven (MAXR) outstanding CDCS SSC requests.	None.	TAF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
n.nnn KILO CDCS REQUESTS FROM TASKS.	Total number of CDCS SSC requests issued by tasks. The number does not include terminate requests which are blocked by TAF.	None.	TAF
n.nnn KILO ITASK RELOADS.	Upon transaction facility termination, this message indicates how many reloads of ITASK occurred. Because ITASK is a central memory resident task, if it makes a fatal error, it must be reloaded.	Inform site analyst.	TAF
n.nnn KILO LOCK REJECTS-filenam.	TAF CRM could not lock records in filenam for a task because the locks parameter on the CRM statement for that file was not large enough.	The site analyst should consider increasing the locks parameter on the CRM statement if lock failures occur frequently.	TAF
n.nnn KILO LOCKS-filenam.	TAF CRM processed nnnn locks for file filenam.	None.	TAF
n.nnn KILO NO COMMUNICATION BLOCKS.	Upon transaction facility termination, this message indicates the number of times a communication block was needed and was not available.	Inform site analyst. The site analyst might want to consider using the K.CMB command during TAF initialization.	TAF
n.nnn KILO NO FL FOR TASK LOAD.	Upon transaction facility termination, this message indicates the number of times TAF was unable to obtain enough memory to load tasks.	Inform site analyst. The site analyst might want to consider using the K.MFL command during TAF initialization.	TAF
n.nnn KILO NO SUBCONTROL POINTS.	Upon transaction facility termination, this message indicates the number of times a subcontrol point area was needed and was not available.	Inform site analyst. The site analyst might want to consider using the K.SCP command during TAF initialization.	TAF
n.nnn KILO OPEN REJECTS-filenam.	TAF CRM could not open file filenam for a task because the users parameter on the CRM statement is not large enough.	The site anlayst should consider increasing the users parameter on the CRM statement if open failures occur frequently.	TAF
n.nnn KILO OPENS-filenam.	TAF CRM processed nnnn opens for file filenam.	None	TAF
n.nnn KILO STORAGE MOVES OF TASKS.	Upon transaction facility termination, this message indicates the number of subcontrol points that were moved. This is done to make more effective use of the field length of TAF.	None.	TAF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
n.nnn KILO TAF FL INCREASES.	Upon transaction facility termination, this message indicates the number of times TAF increased its field length while trying to satisfy an internal request for a block of memory. This internal request is often issued to obtain memory in which to load a task.	None.	TAF
n.nnn KILO TASK RECALLS.	Upon transaction facility termination, this message indicates the number of times tasks were put in recall. This is an internal statistic over which the applications programmer has little control.	None.	TAF
n.nnn KILO TASK RELOADS.	Upon transaction facility termination, this message indicates how many reloads of tasks occurred. TAF reloads tasks that have fatal errors if one of the following is true. - The task is central memory resident. - There are queued transactions waiting to use the task.	Inform site analyst.	TAF
n.nnn KILO TASK ROLLOUT COMPLETES.	Upon transaction facility termination, this message indicates the number of times TAF rolled out tasks. This statistic excludes the number of times the rollout operation did not complete.	None.	TAF
n.nnn KILO TASK ROLLOUT STARTS.	Upon transaction facility termination, this message indicates the number of times TAF initiated a rollout operation. Rollout includes writing a task to disk and releasing the task subcontrol point space. In some cases, the task is allowed to stay central-memory-resident for a while before it is rolled out.	None.	TAF
n.nnn KILO TASK ROLLOUTS BY TAF DM.	Upon transaction facility termination, this message indicates the number of times a rollout was initiated for a task as a result of the TAF data manager detecting a locked record. This occurs when the TAF data manager is unable to proceed because a record has been locked for a certain amount of time. The rollouts are initiated by a request from the TAF data manager.	None.	TAF
n.nnn KILO TASK STARTS BY TAF D.M.	Upon transaction facility termination, this message indicates the number of times tasks were activated as a result of a TAF data manager event, such as a locked record.	None.	TAF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
n.nnn KILO TRANSACTION ABORTS.	Upon transaction termination, this message indicates how many transaction tasks have aborted.	Data base administrator may have to correct data base to account for transactions.	TAFNAM2 TAFTS2
n.nnn KILO TRANSACTIONS PROCESSED.	Self-explanatory.	None.	TAFNAM2 TAFTS2
xxxxxxx NOT DEFINED.	The terminal name xxxxxxx is not defined in the NETWid file.	Consult TAF data base administrator.	TELEX
filenam NOT FOUND.	The user specified the P option on the KTSDMP control statement; one of the two files was not in the user's permanent file catalog or a PFM error occurred.	Specify a different file name or inform a site analyst.	KTSDMP
xxxxxx NOT INITIALIZED BY TOTAL. STATUS IS yyyy.	An error was encountered on the Total data base.	Regenerate Total data base. Refer to Total Reference Manual for status.	TAF
nnnn PER CENT CPU USAGE.	Summary message indicating CPU usage by the transaction subsystem.	None.	TAFNAM2 TAFTS2
nnn TASKS NOT LOADED INTO ECS.	An insufficient amount of ECS was available to load all tasks. The nnn field is the number of tasks not loaded.	Check ECS requested and reinitialize with more ECS if appropriate.	TAF
xxxxxxx TURNED OFF.	Terminal xxxxxxx has been turned off with the K.OFFTERM command and is currently not available for use.	Retry at a later time.	TELEX

The sample deck structures in this appendix demonstrate the use of various parameters which are available. The default input and output files can be used to reduce the number of control language statements needed to bring about a compile, load, and LIBTASK sequence, because the default binary output files are the binary input files for subsequent processors.

For example:

```
COBOL5,ANSI=77LEFT,UC1.
LOAD(LGO)
NOGO(LGOB)
LIBTASK,P=RLTASKL,TT.
```

It is assumed that each of the following decks is preceded by job and USER statements. Some sites may require a CHARGE statement.

COMPASS TASK DECK STRUCTURE

A sample deck which assembles a COMPASS application program and loads it onto a task library, RLTASKL, is illustrated in figure C-1.

```
ATTACH,OPL/UN=usernum.
COMPASS.
LDSET(LIB=TRANLIB)
LOAD(LGO)
NOGO(LGOB)
LIBTASK(P=RLTASKL,B=LGOB,TT,I=0)
7/8/9
        IDENT   TASK
        ENTRY   TASK
OPL     XTEXT   COMKMAC
COMBL   BSS     69
            .
            .<------ COMPASS task appears here
            .
        END
6/7/8/9
```

Figure C-1. COMPASS Deck Structure

NOTE

The system OPL must be attached for COMKMAC to be called in the COMPASS program. Consult site personnel for the appropriate control statement.

FORTRAN TASK DECK STRUCTURE

A sample deck which compiles a FORTRAN application program and loads it onto a task library, RLTASKL, is illustrated in figure C-2.

```
FTN.†
LDSET(LIB=TRANLIB)
LOAD(LGO)
NOGO(LGOB)
ATTACH(RLTASKL/M=W)
LIBTASK(P=RLTASKL,TT,I=0)
7/8/9
OVERLAY(ABS,0,0)
PROGRAM TASK2††
    .
    .<------ FORTRAN task appears here
    .
6/7/8/9
```

†The SEQ,TS, and OPT=0 options are not allowed.
††TASK2 is the name of the task.

Figure C-2. FORTRAN Deck Structure

COBOL 5 TASK DECK STRUCTURE

A sample deck which compiles a COBOL application program and loads it onto a task library, RLTASKL, is illustrated in figure C-3.

```
COBOL5,ANSI=77LEFT,UC1.
LDSET(LIB=TRANLIB)
LOAD(LGO)
NOGO(LGOB)
LIBTASK(P=RLTASKL,B=LGOB,TT,I=0)
7/8/9
    .
    .<------ COBOL task appears here
    .
6/7/8/9
```

Figure C-3. COBOL Deck Structure

NOTE

Under COBOL, the PROGRAM-ID name is used for the task name. Only alphanumeric names can be used as task names.

Unless otherwise specified in xxJ, all files are assumed to be DI (844) resident.

FILE DESCRIPTIONS

xxJ

This indirect access file identifies the journal files for a specific data base (xx) and provides the user index and user number of the data base.

TASKLIB

System task library. This direct access file contains the application program tasks for a system and is defined under the transaction subsystem user number.

JOUR0

System journal file. The transaction subsystem writes journal entries for all transactions to this direct access file.

xxTFIL

TAF data manager trace file. One direct access trace file for each data base (xx) contains a record of all writes to each file to be traced as a result of transactions. (Refer to the TAF/TS version 1 Data Manager Reference Manual.)

xxERPF

TAF data manager error recovery pool file for data base xx. This direct access file contains copies of records which could not be written to their original place of residency because of a mass storage error. (Refer to the TAF/TS version 1 Data Manager Reference Manual.)

xxJORn

Optional direct access journal files for each data base (xx) for task journalizing of transactions.

xxTASKL

User-specified task library for data base xx. This must be a direct access file.

TAFTS

Procedure file containing control statements for TAF using a time-sharing interface.

NETWid

Terminal network file; id equals machine id. This must be a direct access file.

SIMFid

Terminal network file for stimulator. This must be a direct access file.

TCF

The TAF configuration file. An indirect access file which associates data base names with the desired data manager. It is used to specify which data bases and data managers are currently active (able to be used within TAF).

FILE DEFINITION BY USER NAME

Files used in the transaction subsystem can be divided into three general categories.

- Files defined under the transaction subsystem user number. There is a default user number, with password, and a user index defined for the transaction subsystem. This requires reassembly to change. (Refer to the NOS Installation Handbook for more information.)
- Files defined under a data base user number. These files must be permitted in write or modify mode to the transaction subsystem user number.
- Files defined under system user number.

In the following lists, the letters xx signify the two-character data base name. Files are listed according to the user number under which the files must be defined or are defined by the subsystem.

Associated with Transaction Subsystem User Number

JOUR0

TASKLIB

TCF

xxJ

Associated with Data Base User Number	
xxTFIL†	
xxERPF†	
xxJOR1	
xxJOR2	
xxJOR3	
xxTASKL	
xx†	(File containing data definition information for data base xx)
xxaaaa . . . xxzzzz	(Data files in data base xx)
Dxxiiii†	(Dual-recorded files for data base xx)

Associated with System User Number
TAFTS
NETWid
SIMFid

RESERVED FILES

In addition to the files described earlier in this section, the following files are used by TAF. No task or procedure file called by TAF can attach or use any file having one of these names.

KTSROLL

OUTPUT

RECOVR

SCRA through SCRZ

SCR1 through SCR5

TROB

MAP

ZZZZZ.. (File names starting with 5 Z's)

†TAF data manager only.

TAF prevents situations from occurring in which the execution of a task is blocked because it is impossible under the circumstances to obtain the needed central memory (CM) space for the task, whether for an initial load or a memory increase. Several terms will be defined to aid in the description of how potentially blocked tasks are detected:

- Minimum size of the total task area

  The total task area is the CM in which TAF will schedule tasks. It includes the space occupied by CM resident tasks. The area extends from the first word address into which tasks can be loaded to the maximum field length (MFL) of TAF. The size of this area fluctuates if the TAF/CRM data manager is used. The minimum size of this area occurs when the TAF/CRM buffer takes on its maximum size.

- Minimum size of the transient task area

  The transient task area is that portion of the total task area not occupied by CM resident tasks. The minimum size of this area is defined as the minimum size of the total task area minus the sum of the MFL's of all CM resident tasks.

One or more tasks will be considered to be potentially blocked under the following circumstances:

- If the initial field lengths of the CM resident tasks are so large that they all could not be loaded together.
- If the sum of the MFL's for the CM resident tasks exceeds the minimum size of the total task area.
- If the MFL of a non-CM resident task exceeds the minimum size of the transient task area.

The following are the times when potentially blocked tasks are detected and the actions which are taken:

- At TAF initialization. Potentially blocked tasks result in TAF being aborted.
- When TAF is informed, via the LIBTASK TT option, of the update of a task library attached by TAF and the library contains potentially blocked tasks, the update will be rejected.
- When an attempt is made to reduce the MFL of TAF via the K.MAXFL command to a value which results in potentially blocked tasks, the command will be rejected.

The above checking assumes that the maximum field length of TAF is set (by default, via K.MFL, or via K.MAXFL commands) to a value which can be attained.

INSTALLATION OVERVIEW F

Table F-1 is provided to aid the systems analyst and data base administrator in performing TAF installation. It is intended to be a chronologically ordered outline of the procedures detailed in the respective manuals. The complete name and publication number of each entry in the Manual column follows the table.

TABLE F-1. INSTALLATION OVERVIEW

Area	Step	Manual	Manual Keyword
1. Install NOS		NOS Install NOS Maintenance	
2. Install Communications Subsystem		NOS Install NOS Maintenance	
3. Install Product Set-COBOL, etc.		NOS Install	COBOL 5 FORTRAN 4 FORTRAN 5 COMPASS
4. Modify/create TAF subysystem	1. Evaluate installation parameters. Some items of interest at initial installation are: • TAF username • TAF user index • TAF password • TAF charge, project numbers • TAF communication interface - 2550's or multiplexers • Number of terminals	NOS Install	Transaction Facility
	2. Modify TAF on deadstart media if required, or	NOS Install NOS Reference	Transaction Facility GTR, LIBEDIT, LIBGEN
	3. Modify TAF on a running NOS system.	NOS Maintenance	SYSEDIT
	4. Install TAF username and charge.	NOS Install NOS Maintenance	TAF installation parameters MODVAL, PROFILE
	5. Install procedure file to initiate TAF.	NOS Install	TAF
	6. Install automatic initiation of TAF upon deadstart if desired.	NOS Install	IPRDECK
5. Install Transaction Application	1. Assign a team to do data base design. TAF interfaces to CDCS, CRM, TOTAL, and the TAF data manager. Areas in data base design that require examination include: • Elements to be accessed. • Relationships between the elements. • Data manager to be used. • File, record and element types to be used to store the data. • Data formats for the tasks. • Mass storage of the data base.	TAF Reference TOTAL Reference AAM Reference TAF Data Mgr Ref TAF/CRM Ref TAF Users Guide CDCS Reference	

TABLE F-1. INSTALLATION OVERVIEW (Contd)

Area	Step	Manual	Manual Keyword
	• Specific processing to be assigned tasks and the relationships between tasks. For TAF, tasks involved with I/O should be small since the tasks will occupy memory for some time. Tasks that are often used and run quickly may be made core resident. Tasks that are often used should be reusable to avoid task loading.		
	• Data base recovery.		
	2. Assign a team to do the communications interface. Areas to examine are:		
	• TAF terminal I/O requests	TAF Reference	
	• Message formats, terminal operator interactions		
	• Recovery of terminal communications		
	• Interface specifications with the tasks's data management activities		
	• NOS Stimulators	NOS Maintenance	
	3. Install a prototype application using the following steps. As you learn more about the capability and performance trade-offs, revise your design. Use stimulators to test the design under a large terminal load.		
	4. Describe and install the data base using the appropriate data manager.		
	• CDCS	TAF Reference CDCS Reference	xxJ files
	• CRM	TAF Reference TAF CRM Reference	xxJ files
	• TOTAL	TAF Reference TOTAL Reference	xxJ files
	• TAF	TAF Reference TAF Data Manager	xxJ files
	5. Describe and install terminal network files.		
	a. Install TAF's NETWid file	TAF Reference NOS Maintenance	Network files Network Description files
	b. Install terminal users in NOS validation files	NOS Maintenance	MODVAL, PROFILE
	6. Modify procedure file that initializes TAF for special preprocessing procedures, such as preparing for recovery.	NOS Install	TAF

TABLE F-1. INSTALLATION OVERVIEW (Contd)

Area	Step	Manual	Manual Keyword
	7. Write application tasks.	TAF Reference	
	• COBOL 5	COBOL 5 Ref. COBOL 5 User's Guide	
	• FORTRAN 4	FORTRAN 4 Ref.	
	• FORTRAN 5	FORTRAN 5 Ref.	
	• COMPASS	COMPASS Ref. NOS Reference Vol. 2	
	8. Modify ITASK to map transaction codes to their associated tasks. Code for handling system transactions and error processing may be desired depending upon application.	TAF Reference NOS Install	ITASK TAF
	9. Modify other TAF furnished system tasks if desired.	TAF Reference	System Tasks
	• LOGT - terminal logout request		
	• MSABT - task		
	• SYSMSG - TAF system messages		
	• Interface specifications with the tasks data management activities		
	10. Build data base task libraries.	TAF Reference	LIBTASK
	11. Build TAF system library.	NOS Install TAF Reference	TAF LIBTASK
	12. Build data base journal files XXJ to give TAF username for data base.	TAF Reference TAF/CRM Reference	XXJ files XXJ files
	13. Build the TAF configuration file (TCF).	TAF Reference	
6. Test the transaction application			
7. TAF operational commands	Initialize TAF from computer console Status TAF from computer console Terminate TAF from computer console	NOS Operator Guide	Transaction Subsystem
8. Modify performance of TAF and application	1. TAF internal design	NOS IMS	TAF
	2. TAF installation parameters	NOS Install	TAF
	3. Terminal stimulations		
	• STIMULA	NOS Maintenance	STIMULA
	• NSTIM	NOS Maintenance	NSTIM
	• ASTIM	NOS Maintenance	ASTIM

Manual	Publication Title	Publication Number
AAM Reference	CYBER Record Manager Advanced Access Methods Version 2 Reference Manual	60499300
COBOL 5 Reference	COBOL Version 5 Reference Manual	60497100
CDCS Reference	CYBER Database Control System Version 2 Reference Manual	60481800
COBOL 5 User's Guide	COBOL Version 5 User's Guide	60497200
COMPASS Reference	COMPASS Version 3 Reference Manual	60492600
FORTRAN 4 Reference	FORTRAN Extended Version 4 Reference Manual	60497800
FORTRAN 5 Reference	FORTRAN Version 5 Reference Manual	60481300
NOS Install	NOS Version 1 Installation Handbook	60435700
NOS Maintenance	NOS Version 1 System Maintenance Reference Manual	60455380
NOS Operator	NOS Version 1 Operator's Guide	60435600
NOS Reference	NOS Version 1 Reference Manual Volume 1	60435400
	NOS Version 1 Reference Manual Volume 2	60445300
TAF/CRM Reference	TAF/TS Version 1 CYBER Record Manager Data Manager Reference Manual	60456700
TAF Data Manager	TAF/TS Version 1 Data Manager Reference Manual	60453100
TAF Reference	TAF/TS Version 1 Reference Manual	60453000
TAF User's Guide	TAF/TS Version 1 User's Guide	60436500
TOTAL Reference	TOTAL - CDC Reference Manual Version 2	76070300

INDEX

Accessing transaction subsystem 1-3; 11-1

Batch
  Interface 1-1; 5-1
  Subsystem interaction 5-1
BEGIN request 2-14
Binary I/O 12-1
Blocked tasks E-1
BTRAN request 5-1
BWAITINP request 2-14

CALLRTN request 2-4
Call tasks 2-5
CALLTSK request 2-5
CDCS
  Access by tasks 3-1
  Data manager concurrency 1-1
  Interaction with TAF 1-2; 3-1
  xxJ files 4-3
CEASE request 2-16
Chain 2-4
Change CMDUMP request values 6-1
Change I/O limit 2-14
Character sets A-1
CMDUMP request 6-4
COMKCBT 12-1
Communication block
  Defined 2-1
  Dump 6-5
  Load overflow data 2-9
  Release overflow data 2-11
  TAF-originated transactions 10-2
Completion processing 2-16
Control bytes 12-1
Create network file 8-2
Create task library 9-4
Create xxJ file 4-4
CYBER Record Manager data manager 1-1; 4-2

Data base journal files 4-4
Data manager
  Concurrency 1-1
  Requests 2-3
  xxJ file 4-1,2,3
Delete tasks from task library 9-5
Describe configuration 8-1
Determine if terminal is logged in 2-18
Determine subsystem for task 2-14
Diagnostics B-1
DIAL directive 8-1
DSDUMP request 6-1
Dump
  Communication block 6-5
  Data buffers 6-1,2,4,5
  Exchange package 6-1,2,4,5
  TAF field length 6-5
  Task field length 6-3,5

ECS 1-3
EDT 4-1
Element descriptor table 4-1
End task 2-16
Entering transaction subsystem 1-3; 11-1
Entry header, journal file 4-4
Error messages B-1
Execute named task 10-4
Extended core storage 1-3
Extract bit string from word 2-19
EXTRACT request 2-19

Field length changes 2-3; 9-3
File
  Definitions D-1
  Descriptions D-1
Files
  Journal 4-4
  JOUR0 4-4
  Network 8-1
  Pool 4-1
  Procedure 4-1
  Reserved D-2
  System 4-1
  TCF 4-6
  Trace 4-1
  xxERPF 4-1
  xxJ 4-1,2,3,4
  xxJORn 4-4
  xxTASKL 9-1
  xxTFIL 4-1

Global task dump limit 6-5
GTDL 6-5

Ignore task in replacement file 9-3,5
IIO request 2-14
Implement application 1-3
Initial task 1-3; 10-1,2; 12-2
Initiate binary output (0007) 12-1
Initiate new transaction chain 2-5
Input/output
  Limit 2-14
  Requests 2-6
Input overflow 2-9,11
Insert bit string into word 2-19
INSERT request 2-19
Installation Overview F-1
Interrogate TST 2-11
ITASK 1-3; 10-1,2; 12-2
ITL request 2-16

JOUR0 file 4-4; D-1
Journal files 4-4
JOURNL request 4-4

KDIS 10-3
K display commands 2-17; 6-1,4,5
K.DSDUMP command 6-4
K.DUMP command 6-5
K.DUMPLIM command 6-5
KPOINT request 2-17
KTSDMP utility/request 6-5

Libraries 9-1
LIBTASK utility/control statement 9-1
LOADCB request 2-9
Load overflow blocks 2-9
LOGIN request 2-18
Log in terminal to TAF 1-3; 11-1
Log out task 2-17-; 10-3
Log out terminal from TAF 2-17
LOGT request 2-17
LOGT system task 10-3

Macros Inside front cover; 10-1
Manipulate user area of TST 2-12
Memory
    Dump requests 2-3; 6-1
    Dumps 6-1
    Status/Control requests 2-3
Message abort 10-3
Message size limits 11-2
Modes of operation 11-1
Move data from terminal to task 2-14
MSABT 10-3; 12-2
Multimainframe 1-4
Multiplexer directive 8-1

NETWid file D-1
Network
    File origination 8-2
    Files 8-1
NEWTRAN request 2-6

OFFTASK 10-4; 12-2
On-line transaction processing 1-1
Operator interface 7-2
Output requests 2-6

Port number 8-1
Priority, task scheduling 9-4
Procedure files 4-1; 7-1

Queuing
    Forced 9-4
    Normal 9-4

RA+1 calls 2-14
Read data from terminal 2-7
Read security 8-2
RECALAL request 2-16
Record data on journal file 4-4
Recovery 7-1
Release overflow blocks 2-11
RELSCB request 2-11
Requests
    Data manager 2-3
    Input/output 2-6
    Journal file 2-3; 4-4
    Memory dump 2-3; 6-1
    Memory status and control 2-3
    System 2-3
    TAF executive Inside front cover
    Task
        Control 2-16
        Scheduling 2-3
        Utility 2-18
Reserved files D-2

Sample deck structures C-1
Schedule tasks 2-3
Scheduling priority 9-4
Send message to terminal 2-9
SEND request 2-9
Sense switches 7-1
Set binary input mode (0006) 12-1
Set GTDL 6-5
Set time limit for task 2-16
Special characters 11-2
Specify communication block location 2-14
Start transaction using batch job 5-1
Status of transaction 2-6
STRAN macro 10-1
Subcontrol points 1-1,3
Submit batch job using task 5-3
SUBMT request 5-3
Subtransaction codes 10-1
Suspend processing 2-18
SYSMSG 10-4; 12-2
System
    Files 4-1
    Journal files 4-4
    Requests 2-3
    Task library 9-1
    Tasks 10-1

Tables
    Element descriptor (EDT) 4-1
    Terminal status (TST) 8-3
TAF
    Configuration file 4-6; D-1
    Data manager 4-1
    Executive requests Inside front cover; 1-3
    Installation F-1
    Originated transactions 10-2
    Processing advantages 1-1
TARO request 2-12
Task
    Blockages E-1
    Call 2-4,5
    Chain 2-4
    Control requests 2-16
    Defined 1-1; 2-1
    Dump 6-1
    End 2-16
    Library
        Create 9-4
        Defined 9-1
        Delete task from 9-5
        Ignore task in replacement file 9-5
        Update 9-5
    Queuing 9-4
    Pause 2-16,18
    Residency 9-1
    Scheduling priority 9-4

Scheduling requests 2-3
Time-originating 10-1
Type 9-1
Utility requests 2-18
TASKLIB 9-1; D-1
TCF file 4-6; D-1
Terminal
Control 12-1
Interface 11-1
Network description file 8-1
Status table
Figure of 8-3
Interrogate 2-11
Manipulate user area 2-12
TIMCNT macro 10-1
Time-originating tasks 10-1
Time-sharing command subset 11-2
Total data manager 1-1; 4-3,6
TPSTAT request 2-14
TRAN macro 10-1
TRANACT subsystem 11-2
TRANCHK request 2-6
TRANS 10-1
Transaction
Codes 10-1
Defined 1-1
Executive 2-1
Processing 1-3
Subsystem
Entering 1-3
Interaction 5-1
TAF-originated 10-2
TRANT 10-1
TSIM request 2-11
TST (refer to Terminal status table)
TTOT 10-1

Updating task library 9-5

VALNET statement 8-2
VALNET validation 8-2
Validate terminal network description file 8-2

WAIT request 2-18
Wait for terminal input 2-7
WAITINP request 2-7

XTASK 10-4; 12-2
xxERPF file 4-1
xxJ files
CDCS 4-3
Creating 4-4
CRM data manager 4-2
General description 4-1; D-1
No data manager 4-3
TAF data manager 4-1
Total 4-3
xxJORn file 4-4; D-1
xxTASKL file 9-1; D-1
xxTFIL file 4-1; D-1

COMMENT SHEET

MANUAL TITLE: CDC TAF/TS Version 1 Reference Manual

PUBLICATION NO.: 60453000 REVISION: F

NAME:_____

COMPANY:_____

STREET ADDRESS:_____

CITY:_____ STATE:_____ ZIP CODE:_____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

TAPLE

STAPLE

OLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

OLD

FOLD

External A/D

TITLE: CDC TAF/TS Version 1 Reference Manual

PUBLICATION NO.: 60453000
REVISION: F

CONTROL DATA CORPORATION
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

REASON FOR CHANGE:

DATE: 12-05-80

Manual revised to reflect NOS 1.4 at PSR level 530, and to make technical and editorial corrections. Support of TAF/CDCS interface is included. This edition obsoletes all previous editions.

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION